Novel central pattern generator elements for autonomous modular robots

Thesis submitted to the Universidad Autónoma de Madrid for the degree of
Doctor en Ingeniería Informática y Telecomunicación

Author
Fernando Herrero Carrón

Supervisors
Prof. Dr. Pablo Varona Martínez
Prof. Dr. Francisco de Borja Rodríguez

Department
Departamento de Ingeniería Informática

Abstract

Central Pattern Generators are neural circuits that are found in living beings, from simple animals like mollusks, up to higher mammals and human beings. They are responsible for the control of rhythmic motor activities including walking, breathing and chewing among others. Their patterns of activity are robust yet flexible, meaning that some intrinsic, measurable properties of the rhythm are kept invariant, while at the same time modulation of the rhythm is possible, for instance in frequency or amplitude. For all these reasons, CPG models have been used as locomotion controllers in different types of robots. In particular, the modular nature of CPGs has fit very well in the specific field of robotics dealt with in this thesis: that of modular robots.

Following recent research on living CPGs, we propose a new concept of bio-inspired controller for a modular worm-like robot. The central biological principles from which this thesis draws inspiration are: rich intrinsic dynamics for neurons and synapses, mutual inhibition in non-open topologies and winnerless competition dynamics, all of which are characteristic to living CPGs.

In this thesis we first introduce and study the neuron and synapse models that will later be used to build different CPG circuits. We analyze the relationship between individual parameters and neural and synaptic activity. Using these models, we build a simple oscillator to control a single module of the robot. Again, we study its behavior and the influence of different control parameters. Finally, we solve the problem of decentralized, autonomous rhythm coordination. With the right connectivity topology and dynamics, each module negotiates with its neighbor to establish a rhythmic oscillation and an overall frequency and phase difference. For this, we propose four different inter-modular connectivity patterns. These patterns are scalable and, just by repetition, CPGs with an arbitrary number of modules can be constructed.

The result is a set of new bio-inspired strategies that can be used to build effective CPG controllers for autonomous modular robots. The right combination of intrinsic dynamics and topological connectivity is the key to adaptability and self-organization of the locomotive rhythm. We show that the dynamical properties of the models make the proposed modular oscillator more resilient to noise than other state of the art works. Then, we couple our oscillator to different simulated servomotors without changing its parameters, and show that it can adapt its frequency of oscillation over some orders of magnitude, depending on servo responsiveness. Finally, we show that the four proposed connectivity patterns generate the desired rhythm, and CPGs

built this way effectively generate forward locomotion in a real robot.

Resumen

Los Generadores Central de Patrones son circuitos neuronales que se encuentran en los seres vivos, desde animales simples, como los moluscos, hasta los mamíferos superiores y los seres humanos. Ellos son los responsables del control rítmico de actividades motoras como caminar, respirar y masticar entre otros. Sus patrones de actividad son robustos y flexible, es decir, que algunas propiedades intrínsecas medibles en el ritmo se mantienen invariantes, mientras que al mismo tiempo se puede modular el ritmo, por ejemplo en frecuencia o amplitud. Por todas estas razones, los modelos de CPG se han utilizado como controladores de locomoción en diferentes tipos de robots. En particular, la naturaleza modular de los CPGs ha encajado muy bien en el campo específico de la robótica abordado en la presente tesis: el de robots modulares.

Partiendo de investigaciones recientes sobre CPGs vivos, se propone un nuevo concepto de controlador bio-inspirado por un robot modular con forma de gusano. Los principios biológicos centrales en los que esta tesis se inspira son los siguientes: dinámica intrínseca rica de neuronas y sinapsis, inhibición mutua en topologías no abiertas y dinámica de competición sin ganador, todos los cuales son característicos de los CPGs vivos.

En esta tesis, en primer lugar introducimos y estudiamos los modelos de neuronas y sinapsis que más tarde se utilizarán para construir diferentes circuitos. Analizamos la relación entre parámetros individuales y las actividades neuronal y sináptica. Usando estos modelos construimos un oscilador simple para controlar un módulo individual del robot. Una vez más, se estudia su comportamiento y la influencia de los diferentes parámetros de control. Por último, se resuelve el problema de la coordinación descentralizada y autónoma del ritmo. Con la topología y la dinámica apropiadas, cada módulo negocia con su/s vecino/s para establecer una oscilación rítmica y una frecuencia y diferencia de fase globales. Para ello, proponemos cuatro patrones de conectividad modular diferentes. Estos patrones son escalables y, simplemente por repetición, se pueden construir CPGs con un número arbitrario de módulos.

El resultado es un conjunto de nuevas estrategias bio-inspiradas que se pueden utilizar para construir controladores CPG efectivos para robots modulares autónomos. Una combinación apropiada de dinámica intrínseca y conectividad topológica es la clave para la adaptabilidad y la auto-organización del ritmo de locomoción. Demostramos que las propiedades dinámicas de los modelos proporcionan mayor tolerancia al ruido a nuestro oscilador que otros modelos en el estado del arte. Luego, acoplamos nuestro oscilador a diferentes servomotores simulados sin cambiar sus parámetros, y demostramos

que puede adaptar su frecuencia de oscilación varios órdenes de magnitud, en función de la capacidad de respuesta del servo. Por último, demostramos que los cuatro patrones de conectividad propuestos generan el ritmo deseado, y que los CPGs construidos de esta manera generan una locomoción efectiva en un robot real.

Contents

Part I

Introduction

1 Introduction

Effective locomotion is an ability inherent to the animal kingdom. Throughout evolution, life has put into test many different designs to solve the problem. As a result, the present landscape of living forms is a compendium of tested and validated locomotion solutions. Not surprisingly, there are generic mechanisms that solve similar solutions in different contexts: phenotipically distant species use the same strategies to achieve similar goals. Through the study of the nervous system commanding locomotion, we can unveil these strategies that can be applied to design novel robotic paradigms.

There is an increasing amount of new results of motor control research in living neural systems that remain unexplored in the context of bio-inspired robot locomotion (Grillner, 2006; Marder and Bucher, 2007; Smarandache et al., 2009). Of particular interest to robotics are the studies regarding *Central Pattern Generator* circuits (Marder and Bucher, 2001; Selverston et al., 2000). CPGs are neural networks that generate rhythmic activity to control motor neurons, and are involved in motion that require periodicity, robustness and/or precision. CPGs are autonomous in the sense that they do not need external input to produce a rhythm. However, sensory signals modulate CPG activity in order to adapt to external conditions.

CPGs of invertebrates are the best known neural circuits in neuroscience research (Selverston and Moulins, 1987; Getting, 1989; Arshavsky et al., 1991; Grillner, 2006). Recent studies in living CPGs have shown that these circuits (i) have common connectivity building blocks based on mutual inhibition (Selverston et al., 2000; Huerta et al., 2001; Stiesberg et al., 2007); (ii) have neurons and synapses that exhibit rich dynamics with multiple time scales to swiftly negotiate robust sequential activations (Rabinovich et al., 2006); (iii) display dynamic invariants to preserve rhythms that are simultaneously robust and flexible (Grillner, 2006; Marder and Bucher, 2007; Reyes et al., 2008); (iv) have multiple codes that allow cells to multiplex both neural *messages* and neural signatures, a mechanism that can allow a receiver neuron to identify who the sender cell is (Szucs et al., 2003; Latorre et al., 2006).

In this context, modular robotics provides a flexible platform where differ-

ent locomotion paradigms can be studied (Kurokawa et al., 2008; Yim et al., 2000). Using a number of homogeneous modules one can easily construct different sized robots, reconfigure their topology or assemble completely newly shaped robots. Moreover, modular robots represent a very good starting point for new paradigms. By their very nature, there exist a number of problems that must be solved at different levels of abstraction. For instance: how to code information in one individual module, how to build a single oscillator or how to couple oscillators together. Furthermore, modularity calls for generic principles that will scale well when new modules are added in. That is, there is the need for design patterns that can be reproduced locally in each module, while maintaining the global invariant of effective locomotion.

Neuroscientific CPG knowledge has already been successfully applied to robotic control (Ijspeert, 2008; Degallier and Ijspeert, 2010) focusing on different aspects: for instance, Ayers et al. (Ayers and Witting, 2007) developed a highly realistic motion model of a crustacean limb, while Arena et al. (Arena et al., 2005) developed an artificial neural network to control a hexapod robot; different forms of fin/wing control have been achieved by Chung et al. (Chung and Dorothy, 2009) and Seo et al. (Seo et al., 2010) both of them with an extensive analysis of the convergence and stability of the controllers. Besides these, there has been work done on biped locomotion using CPGs (Manoonpong et al., 2007; Aoi and Tsuchiya, 2006) and modular locomotion (Ijspeert and Kodjabachian, 1999); and there have been different approaches to learning, for instance off-line genetic CPG design (Kamimura et al., 2003) and on-line optimization methods (Crespi and Ijspeert, 2008; Sproewitz et al., 2008).

In most cases, CPG bio-inspiration in robotics uses the scientific knowledge from these circuits that was available more than twenty years ago. Thus, bio-inspiration is often reduced to the use of oscillators implemented with basic, single time-scale limit cycle behavior. While this type of CPG control has proved highly successful, in this paper we argue that the use of novel findings regarding living CPGs can result in more general design strategies for autonomous locomotion in modular robots. We argue that the proposed biological strategies will provide greater flexibility and robustness and lead to more autonomous behavior (Herrero-Carrón, Rodríguez and Varona, 2011).

In this thesis we first describe and explore a neuron and a synapse model with that are both dynamically rich and computationally inexpensive. Then, we study the effect of control parameters on the activity of those models. Afterwards, we design a modular oscillator using the appropriate dynamics and topology. Finally, we explore what principles are effective topology patterns to interconnect a chain of modular oscillators and integrate all these results to control a real robot.

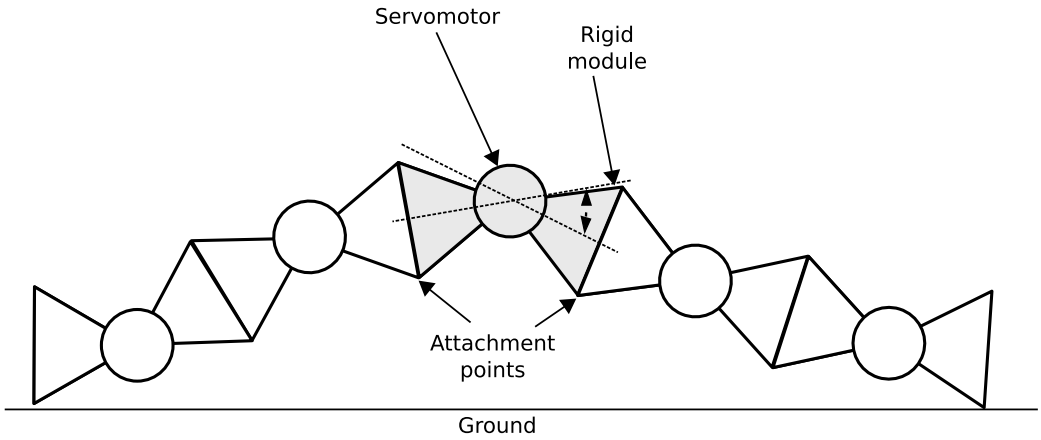


Figure 1.1: General schema of the worm robot. A single module is marked in gray, all other modules are exactly equal to this one.

All of this is demonstrated in a robotic platform designed by González-gómez et al.. This platform is very powerful, in terms of locomotion capabilities, while still being very accessible and easy to control. For simplicity, we have focused on horizontal ground displacement, one of the many locomotion modes this robot is capable of.

The robot, illustrated diagrammatically in figure 1.1, consists of several modules attached side by side through special connection points. Each of these modules consists of two triangle shaped rigid pieces, joint by one vertex of the triangle, and a servomotor controlling the angle between these two pieces. In horizontal locomotion mode, modules are connected sequentially, each of them oscillating on the same plane. One solution to the control problem here posed is undulatory locomotion. Each module must oscillate periodically, synchronized with a given phase lag from the neighboring ones. Thus, the CPG must solve the problem of individual oscillation, global coordination and overall autonomous adaptability.

The choice of this platform has been motivated by its versatility (the reader is referred to (González-gómez et al., 2006)), low cost and ease of construction. The chassis is built of methacrylate panels, assembled by hand in less than one hour. The servos are futaba S3003[1], readily available in any RC store and with an approximate cost of $15 apiece. Finally, there being no wheels, limbs or any other movable parts besides the servos, control of the robot is exclusively a problem of synchronization among modules, a problem that CPG control will solve in a robust and flexible manner.

In the following chapters we will first make an introduction to some gen-

[1] `http://www.gpdealera.com/cgi-bin/wgainf100p.pgm?I=FUTM0031`

eral concepts from *systems theory* (von Bertalanffy, 2006) that will be useful for discussions in later chapters. Then, we will review recent research on living CPGs that provide insight on the mechanisms that contribute to the flexibility and robustness of these circuits. Following this we will review the literature on the application of CPGs to control modular robots. Afterwards, we will make a brief introduction to the methodology of dynamical systems and explain the neuron and synapse models used in the results part. To conclude the introduction, we will explain the mathematical tools used for the analysis of oscillatory signals generated by the CPGs in the different sections of this thesis. In an appendix we discuss some implementation details of the library created to implement the CPG circuits developed in this thesis.

2 A Systems Approach

It is difficult to arrive at a precise, universal definition of the concept *system*. It will suffice for this thesis to describe some ideas relating to systems theory that will be useful to understand other concepts when studying CPGs.

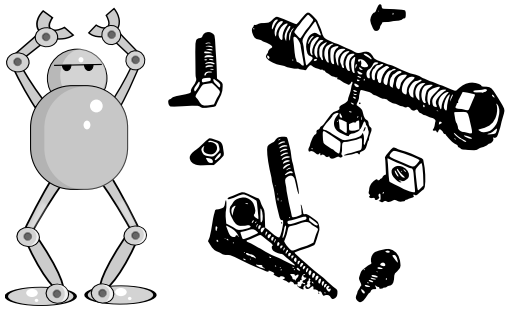
One of the difficulties in defining what systems are, is that they are only indirectly observable through the interaction of their parts. A system is somehow the integration, temporal or permanent, of a set of parts into a whole. This integration happens through stable relationships between the parts. This relationships, in the most general case, include the exchange of energy, matter and/or information. The interaction between the parts defines a virtual boundary that separates the outside of the system from the inside of the system, i.e. those parts interacting from those other parts not in interaction. Some systems, specially biological ones, also have a physical structure that acts as a boundary.

The robot in figure 2.1 is an illustrative example of the concept of a system. It is not only the parts by themselves that make the robot, it is those parts standing in a specific relationship to each other: only the whole of the parts and the relationships between them is meaningful as a robot.

Continuing with the parts of a system, they may also be systems in themselves. That is, the *space*, physical or virtual, enclosed within the boundaries of a system may be subdivided and parceled into further *subsystems*, establishing an organizational hierarchy in this way.

Attending to the nature of the parts of a system, we can say a system is *homogeneous* or *heterogeneous*. Homogeneity means that the parts of the system have all similar qualities, for example similar function or similar structure. Heterogeneity means that some parts possess qualities that other parts do not possess. Observing the natural world, there is a complex equilibrium between both. Looking at living CPGs, we can find heterogeneous groups of neurons interacting among themselves. In turn, these groups are usually aggregates of homogeneous neurons that behave similarly.

If we analyze the nature of the interactions between parts of a system, we can establish further classifications. These interactions can be *linear* or *non-linear*. Linear interactions occur when changes in one part of the



(a) (b)

Figure 2.1: Picture (a) shows a *robot*. Where does the concept *robot* emerge from? Looking at picture (b) we can identify *parts* of a robot, but we cannot affirm that they *are* a robot. A complex system can only be understood as the total result of its parts interacting in a specific way. The difference between pictures (a) and (b) is that all parts in (a) are interacting in such way that a *robot* emerges, while in (b) their relationships do not result in a *robot*.

system evoke proportional changes in another part of the system. Non-linear interactions, on the other hand, produce complicated relationships between causes and effects. Most, if not all interactions between parts of living CPGs are non-linear. Systems where interactions are non-linear are called *complex systems*.

When parts of a system interact they become interdependent, in the sense that changes happening in one part will affect the other interacting parts. We say that those dependent parts are *coupled*. Coupling can be *tight*, if all, or most of all changes in one part affect the others, or *loose*, if only a few changes in one part affect the others.

2.1 Living systems

Even though the number of different systems is almost infinite, some common properties are found among them. This section will describe some basic system processes that living CPGs also share.

Everything that lies outside the boundaries of a physical system is its *environment*. The environment is always changing, and its characteristics may or may not be compatible with the existence of a given system. One key aspect of living systems is that all of them have developed mechanisms to preserve their structure and/or function over time, even in the face of environmental changes. This ability of a system to preserve itself in a changing environment is called *adaptation*.

Adaptation happens when a system has some parts whose function is perception, and some parts whose function is action. Through perception, an adaptive system perceives environmental, and possibly, internal conditions. Through action the system is able to adapt its inner state to match external conditions and ensure further living. These functions of sensing and acting may be as general as can be, and do not necessarily require any complex intelligence or planning of action. For instance, plants grow towards light. They perceive, somehow, the direction from which light is coming, and move and grow in that direction. There is, as far as we know, no global plan on how a plant will grow.

A special case of adaptation is *auto-regulation*. If a sand castle is left to dry, it will fall apart. It requires constant moistening to stand and continue to exist as a castle. Equally, all living beings need to invest energy in their own preservation. The process of auto-regulation is the process by which some or all parts of a system work to ensure that the inner state of the system is kept within safe limits. In living beings, this includes the acquisition of sufficient energy and matter, the disposal of waste, metabolism, etc. It is crucial that the parts responsible for auto-regulation receive information about their performance. In some way, we can think about parts as subsystems within an environment, that need to adapt. In the case of auto-regulation we will generally speak about *feedback*, instead of sensing or perception.

In conclusion, it should be clear by now that the behavior of a system is the result of the behavior of its parts and the nature of their relationships. In order to transfer knowledge of systems in one domain to systems in a different domain, we need to understand the dynamics of the parts of the former and the nature of their relationships, and build abstract models that can be applied to the latter.

2.2 Transfer of knowledge

The study of CPGs is not different from the study of other physical or biological phenomena. The process is an iteration of observation, formulation of hypotheses, experimentation, and testing of hypotheses. Central to this loop is mathematical modeling of the system under study.

2.2.1 The experimental paradigm

The motivations for building models of an existing object are usually either the desire to control it, or the desire to have a replica with the same characteristics that can be used instead of the original one. Let us investigate the process of knowledge transfer between different domains, the role of modeling, and let us put it in a framework of systems theory.

The experimental process begins with an act of observation. We, as humans, perceive a reality, an object, that we want either to control or to replicate. Observation is highly influenced by the nature of the observer. For the observation to deliver any meaningful result, the observer must posses the appropriate sensing or measuring instruments. This is evident from our daily human experience: we have eyes that can perceive light and colors but cannot perceive sounds, we have ears that can perceive vibrations but cannot smell, etc. So the act of observation implies the separation, the highlighting of a specific feature of the object to be observed, namely that which the observer is able to perceive.

Not all features of the object are relevant to the intention of acquiring knowledge about it. Usually only a given aspect of the object will be of interest, or even accessible to the observer. For example, the driver of a car will be interested in increasing or reducing its speed, in steering it, but he will not be interested in its color.

Once the features of interest have been observed, hypotheses may be made about their structure, their function or their relationships. This is the origin of a model. A model is a construction that should have similar properties to those of the studied object. But a model is also an abstraction. The relevant aspects of the object are selected and taken apart from the other aspects, and this is where the transference of knowledge can happen. A model is, in itself, abstract knowledge, made independent from its original domain, and transferable to a different domain.

However, the verisimilitude of a model is restricted by the filtering process of observation. A model can only describe unobserved reality by chance. Therefore, understanding gained from a model of reality does not necessarily match the modeled reality itself. *Experimenting* is then necessary to test the formulated hypotheses. If the test is successful, then the model can be taken in place of the modeled for prediction or study purposes. If not, refinement of both the model and the underlying hypotheses will be necessary, and the process should be iterated again.

Now, we have a plan to transfer biological knowledge from living CPGs to artificial robots (figure 2.2. First, we need to focus on the aspect of CPGs that is interesting to us, namely the dynamics of their parts and their topology. Then, we need to build a model that contains the abstract knowledge that describes that aspect. Finally, we must adapt that model to the robotic domain.

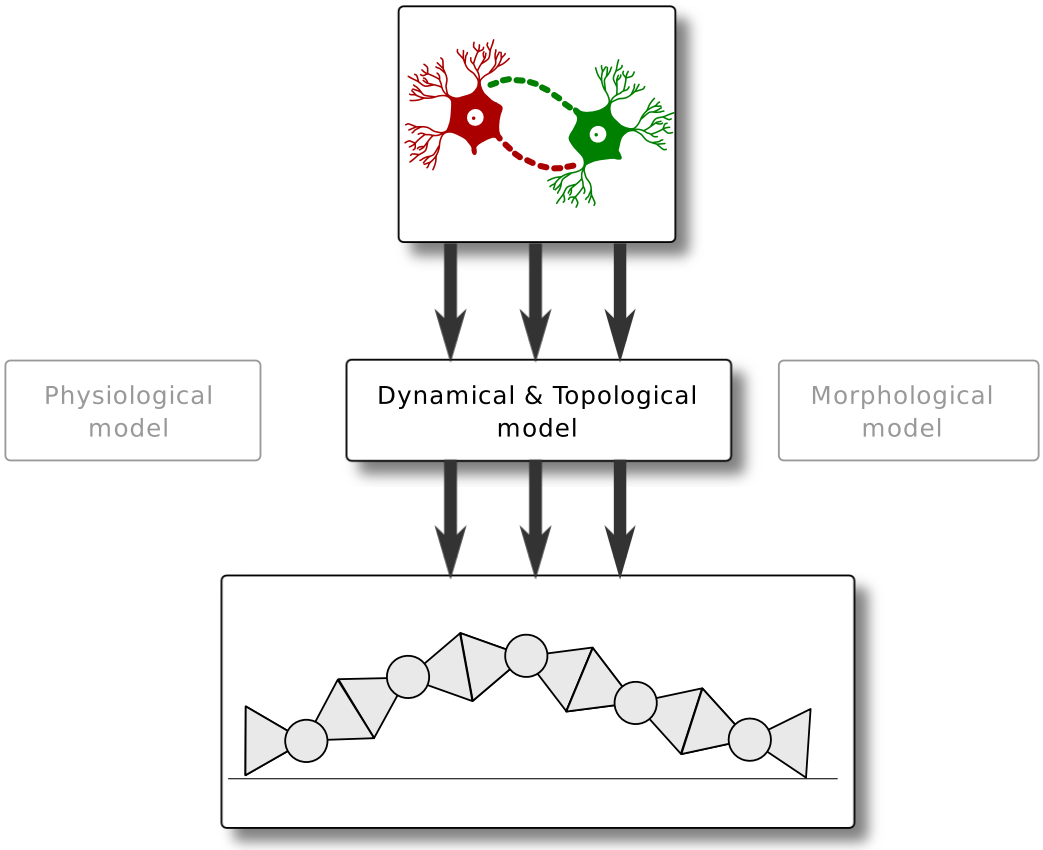


Figure 2.2: Central Pattern Generators are probably the best understood neural circuits. They present interesting dynamical features that engineers would like to replicate in their robots. To transfer knowledge from the biological domain to the robotic domain, domain independent aspects of CPGs must be separated from biologically specific aspects. In particular, the transfer of knowledge can only happen through an abstract model that replicates the dynamical behavior of the CPG components and their interactions.

3 Results on Biological CPGs

Variability is ubiquitous in nature. No two individuals of the same species are exactly the same, and even one single individual grows and changes throughout its life. Quantitatively, individuals will differ in size, weight and possibly also in shape depending on their particular history. However, qualitatively, all members of a species must perform the same functions to survive. Breathing, walking, flying or swimming are qualitatively common activities that each individual performs in a unique quantitative way. This observation applies to individual CPGs controlling those functions as well. Looking at different pyloric CPGs of different individuals of the same species it becomes evident that there is a wide variability from animal to animal (Bucher et al., 2005). However, they all succeed in performing an effective pyloric control. So there must be some inherent characteristic, some quality that is the essence of successful pyloric activation.

We call the essence of the pattern produced by a CPG its *dynamical invariant* (Herrero-Carrón et al., 2010*a*). It has been shown that this concept is indeed present in living CPGs (Reyes et al., 2008). This paper shows that the studied CPGs try their best to keep the relationship between phase and period invariant when artificially modified, even if there is a large variability in phase and period from animal to animal.

Studying the mechanisms underlying dynamical invariants is very promising for robotics research. Clearly, there is something that allows CPGs to grow, evolve, and still preserve the essence of their function despite actual changes. The concept of the dynamical invariant allows for a new design strategy in which the motor control can be achieved in a way that is not specified a priori. The result would be more autonomous CPGs, not only from a synchronization point of view, but also from an adaptation point of view. If we understand the mechanisms behind dynamical invariants, we will be able to construct general controllers independent of the specific details of the robot, or circuits that will adapt to aging and damaging of the robot.

In this chapter we discuss scientific results on the mechanisms known to contribute to the essential flexibility and robustness of CPGs, which may be directly related to the implementation of dynamical invariants.

3.1 Starting, Stopping and Maintenance of Rhythms

Few studies in robotics are concerned with how CPGs are started and stopped. There exist various mechanisms underlying the starting, stopping and sustaining of rhythm in living CPGs.

In cases like breathing or heart beating, the group of neurons responsible for the maintenance of rhythm cannot afford to stop. There are other cases, however, in which a set of neurons needs to be activated and, after some time of activity, deactivated.

In the first case, special neurons called *pacemaker* neurons are responsible for setting the pace at which the rest of the group will oscillate (Selverston et al., 2009). Thanks to biophysical sub cellular mechanisms, they can produce oscillations in isolation, without external input. Through synaptic connections, the activity of pacemakers is *distributed* to other members of the groups.

Pacemaker neurons can usually adapt their burst frequency over a large range. By means of external input and feedback from other neurons in the network, the pacemaker group will be able to decide if the circuit is working properly and, if not, set the appropriate frequency. Some CPGs show redundancy mechanisms in order to maintain the correct rhythm in case of failure. Several neuron models are capable of endogenous oscillation as seen in living CPG neurons: many variations of the bio physically detailed model by Hodgkin and Huxley (Hodgkin and Huxley, 1952); one classical model for bursting activity (Hindmarsh and Rose, 1984) and other recent models with less computational requirements (Rulkov, 2002; Aguirre et al., 2005).

In the second case, initiation and termination are usually caused by external forces. A silent group of neurons may be *recruited* by an external excitatory force. Such is the case in (Arshavsky et al., 1998), where neuron group IN 12 is recruited by neuron group IN 8 and inhibited by neuron group IN 7. This a very good example in which neurons that do not generate a rhythm autonomously contribute to the proper functioning of the CPG.

Interestingly, this CPG presents another important feature. Depending on the intensity of the swimming, an *early* group of neurons is recruited for weak swimming and if more powerful strokes are needed, a neuron group with higher activation threshold will also be recruited. This is the *delayed* group of neurons.

There exists one further mechanism found in some neurons that may initiate activity, known as *post-inhibitory rebound*. This feature is widely observed, in particular again in (Arshavsky et al., 1998). A neuron with this

feature will remain silent while inhibited. If inhibition is released suddenly, the neuron will respond with a spike or burst of activity.

It is believed that the selection of motor programs may be subject to an inhibitory mechanism (Grillner et al., 2005). Each possible motor program is kept under inhibition, until upper control centers decide to activate it. At this point, inhibition is released. By the mechanism of post-inhibitory rebound, a motor program could immediately start upon lifting of inhibition. Note that this is not equivalent to activation by excitation. The difference is that in a normal state, the circuit responsible of the motor program will not spontaneously activate because of noise, since inhibition will keep it forcibly silent.

The mechanism for termination of activity will depend on the goal of the CPG. If neurons in the CPG are not capable of endogenous oscillation, they may just go silent by themselves after activity has been elicited by one of the mechanisms mentioned earlier. If the activity needs to be sustained, then mutual excitatory connections within a group of neurons will keep them firing. This is known as a "pool" of neurons.

So the repertoire for how activity is elicited, maintained and stopped is really ample. However, each mechanism has different subtleties, depending on the underlying cellular characteristics, on whether inhibition or excitation should be preferred, etc.

3.2 Motor Command Coding

Individual CPG neurons display a mainly bursting activity. Motor commands are encoded in some aspect of the neuron's activity, for instance in the frequency of the spikes, or on the precise timing between them. How information is extracted from this activity is not trivial (Brezina et al., 2000; Thuma et al., 2003), but simple mechanisms can be used in robots.

The fact that rhythmic motor commands are encoded using bursting neurons has an advantage over the classical view of CPGs in robotics: the mechanisms that encode motor commands and those used for synchronization are decoupled. In the traditional view, one non-linear oscillator represents a whole CPG. Then, one variable of the oscillator is used as output to the controlled joint and to other oscillators that need to be synchronized. Bursting neurons have the ability to flexibly adapt the timing between bursts and still produce robustly reproducible bursting patterns. With this decoupling, the system gains simultaneously in robustness and flexibility. In addition, some bursting neurons have the ability to encode information relative to their identity and their context and send messages to neighboring neurons, as we discuss in the following section.

3.3 Neural Signatures

Recent experiments have shown that CPG individual cells have neural signatures that consist of neuron specific spike timings in their bursting activity (Szucs et al., 2003). Model simulations indicate that neural signatures that identify each cell can play a functional role in the activity of CPG circuits (Latorre et al., 2006). These signatures coexist with the information encoded in the slow wave rhythm of the CPG which results in a neural signal with multiple simultaneous codes. Readers of this signal (muscles and other neurons) can take advantage of the multiple simultaneous codes and process them one by one, or simultaneously in order to perform different tasks. The sender and the content of the signals can also be used separately to discriminate the information received by a neuron by distinctly processing the input as a function of these codes. These mechanisms can contribute to build dynamical invariants through a self-organizing strategy that includes non supervised learning as a function of local discrimination. Artificial CPG networks built with neurons that display neural signatures allow for a new set of learning rules that include not only the modification of the connections, but also the parameters that affect the local discrimination.

3.4 Topologies

CPGs are known for their flexibility: they generate a robust rhythmic activity with a recognizable pattern, but they can be modulated by external input. The final shape of the rhythm produced by one CPG will depend on many factors: intrinsic properties of individual neurons and synapses, strength and sign of the couplings and network topology, all contribute to the function of the circuit. It is known (Huerta et al., 2001; Stiesberg et al., 2007) that *non-open* topologies maximize the quality of the rhythm produced in terms of flexibility and regularity. A non-open topology is that in which every neuron receives at least one connection from another CPG member, in contrast to an "open" topology, where at least one neuron does not receive synapses from any other CPG member

In the design of an artificial CPG for robot control, it is necessary to take this into account. All neurons within the circuit must receive feedback from the rest of the circuit, so that knowledge about how the CPG is performing is distributed to all units. Of course, not all neurons will process this information in the same way.

Different works, both theoretical and experimental, have studied the role of synapses in the synchronization of neural oscillators. However, this study cannot be decoupled from the properties of the units being connected. Depending on the intrinsic characteristics of neurons, synaptic activity will have

different effects. In fact, different CPGs may adopt different combinations of neural and synaptic properties to achieve the same goal (Prinz et al., 2004). The result is that the CPG designer has a wide variety of mechanisms to choose from. For instance, it is widely accepted that in-phase synchronization can be achieved through excitatory interaction, but it can also be achieved by inhibitory interaction with the appropriate conditions (Wang and Rinzel, 1992). And conversely, anti-phase synchronization is usually considered to happen under inhibition (Wang and Rinzel, 1992; Rowat and Selverston, 1997) but it can be found to happen under excitatory connections as well (Kopell and Somers, 1995).

Usually phase relationship between modules of a CPG is an important dynamical invariant that must be preserved. There exist some important theoretical studies on the mechanisms for phase locking between oscillators. A general study based on a phase model can be found in (Kopell and Ermentrout, 2000). However, it poses strong restrictions upon the coupling between oscillators. More realistic approaches that take into account synaptic dynamics for predicting the type of synchronization among bursting neurons can be found in (Oprisan et al., 2004) and (Elson et al., 2002).

When building complex artificial CPGs, it is difficult to predict how the coupling of neurons within a given topology will affect synchronization. Being able to predict stable phase-locking regimes as in the above mentioned works could help us design complex CPGs, with rich dynamics and guarantee that the resulting phase and frequency relationships will be the desired ones.

3.5 Homeostasis: Self-Regulation Mechanisms

CPG research has also shown the presence of many homeostatic mechanisms to self-regulate and to deal with unexpected circumstances at the cellular and neural network levels and in multiple time scales (Marder and Goaillard, 2006). Dynamical invariants that work in short time scales can be built with the above mentioned mechanisms that involve neuron and synapse dynamics and specific topologies. However other types of invariants working in longer time scales can use mechanisms of adaptation and learning (including sub cellular plasticity) to generate rhythms even in the absence of the input that sustains this rhythm under normal circumstances (Thoby-Brisson and Simmers, 1998). Long scale dynamical invariants arise from self-regulatory mechanism that involve tightly regulated synaptic and intrinsic properties. Models could implement this self regulation through specific synaptic and sub cellular learning paradigms.

Implementing already known sub cellular and network learning mecha-

nisms (Marder and Prinz, 2002), CPGs may easily accommodate the degradation process of a working robot. If a joint loses torque, if pieces begin to wear off, a CPG implementing these mechanisms could adjust its function to reflect the natural evolution of the mechanical parts. Even if some part suddenly stops working or is disconnected from the main body of the robot, the CPG could still keep its original function.

3.6 Summary

Living CPGs have evolved and specialized to perform specific functions. They are involved in the control of rhythmic actions like locomotion, digestion, chewing, etc, that are critical for the preservation of life. Therefore, CPGs are very precise and effective controllers. However, variability abounds: between different animals of the same species, within one single individual in different short term contexts and throughout its lifespan. In all these situations CPGs have proved their adaptability and flexibility, and in all kinds of different conditions, the essence of their function is maintained. We call the essence of the function of a CPG, irrespective of the particular context, its dynamical invariant.

The dynamical invariant concept points towards a generic mechanism to construct autonomous function preserving circuits. Ideally, when translated to robots, we would be able to design artificial neural controllers only defining the core restrictions that they may preserve. Then, using the right set of bio-inspired strategies, a CPG could be built that would autonomously adapt to physically different robots, to short transient perturbations and to long term evolution of the machine.

Living CPGs incorporate adaptation mechanisms that work and interact on several time scales. On a global time-scale, evolution has tried many different solutions, discarded ineffective ones and promoted effective ones. And for a solution to be effective it must include adaptation mechanisms on shorter time-scales. The next temporal frame would be individual lifespans, and the mechanisms responsible for morphological adaptation at a circuit and cellular level. Finally, adaptation on the short term equates to modulation of activity, which relies on the dynamical and topological characteristics of the circuit.

In this chapter we have reviewed some of the literature on biological CPGs. The features discussed all play an important role in short- and long-term adaptability. Short-term adaptation has been explored for robotic control using simple abstractions of the underlying mechanisms. Understanding how the different time-scales interact in biology will surely bring about a break-through in robotics.

4 Central Pattern Generators and Modular Robots

The philosophy behind CPGs has found wide acceptance in the robotics community in general, but specifically in the modular robotics field. Following the reasoning from the previous section, each module of the robot can be endowed with a dynamical controller, and then each module can establish communication channels with its neighbors. Precisely this modular rationale fits perfectly with the CPGs approach.

In this section, we will review the most important robot families that have been an inspiration for this work, focusing on modular robots.

4.1 Modular robotics: a trend is born

Almost any human engineered system is modular in nature, so it would be more precise to write about *homogeneous* modular robots. In fact, the central philosophy behind modular robots is the assembly of simple, similar parts into complex structures. From a control point of view, the modules of a modular robot form a decentralized, flat society of agents that must negotiate and coordinate their actions to perform a global task.

Mark Yim is considered to be the father of modular robotics. In his PhD thesis(Yim, 1994), he addressed two core problems of modular robotics: configuration and control. His work on modular reconfigurability crystallized in his famous robot PolyBot (Yim et al., 2000). As regarding control, he did not claim any bio-inspiration, but his work is worth being mentioned anyway.

His control method was based on look-up tables. For each time step, an index into a table would yield the position values for every joint of the modular robot. This index would be incremented with every step, and when it would reach the end of the table the sequence would just be repeated from the beginning. This mechanism is very simple in computational terms, but clearly locomotion lacks in adaptability. The main focus of successive work on modular robots and CPG control would be on self-organization.

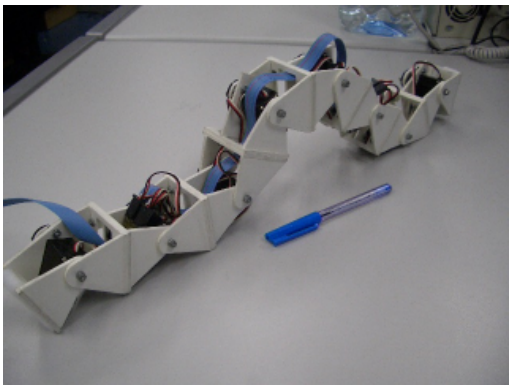


Figure 4.1: Cube Revolutions. Photograph courtesy of Juan González Gómez, UAM

4.2 Universidad Autónoma de Madrid

The first modular robot developed in this university (UAM) was *Cube Reloaded*[1] (González-Gómez and Scalvinoni, 2003), the third in the *Cube* family, preceded by *Cube 1.0* and *Cube 2.0*[2], and followed by the fourth generation *Cube Revolutions*[3] (González-Gómez and Boemo Scalvinoni, 2008) (figures 4.1 and 1.1).

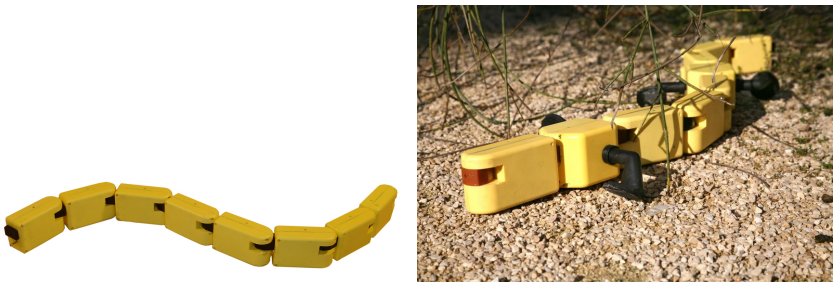
The author investigated the locomotion capabilities of this robot using sinusoidal controllers. The control strategy is static, in the same sense as the look-up tables used by Yim. Bio-inspiration is introduced in the form of rhythmic oscillations with inter-module phase difference. Depending on parameters such as amplitude of oscillation and phase lag, different locomotion gaits can be generated (González-gómez et al., 2006).

The power of this control model lies in the precise controllability of locomotion. In his thesis, González-Gómez precisely characterizes the range of parameters and the gaits they generate. On the other hand, the robot lacks in autonomy, and self-organization capabilities, which also hinders adaptation to unknown terrains.

[1] `http://www.iearobotics.com/personal/juan/doctorado/cube-reloaded/index.html`

[2] `http://www.iearobotics.com/personal/juan/proyectos/cube-2.0/cube.html`

[3] `http://www.iearobotics.com/personal/juan/doctorado/cube-revolutions/index.html`



(a) Amphibot (b) *Salamandra robotica*

Figure 4.2: Photographs courtesy of (a) Biorobotics Laboratory, EPFL and (b) A. Herzog

4.3 École Polytechnique Fédérale de Lausanne

The *École Polytechnique Fédérale de Lausanne (EPFL)*[4] is located in Lausanne, Switzerland. The most prolific author on the subject of modular robotics and CPG control has been Auke Jan Ijspeert. Following previous theoretical models of anguilliform locomotion (Ekeberg, 1993; Ekeberg et al., 1995), he designed a simulation model on which he evolved a CPG using genetic algorithms (Ijspeert and Kodjabachian, 1999).

An extension of the lamprey model lead Ijspeert to his most famous work: Amphibot[5] (figure 4.2a). Amphibot is a chain of modules, with one degree of freedom joints. They all oscillate on the frontal plane of the robot, resulting in an anguilliform locomotion. Extending this model, and using a combination of a genetic algorithm search strategy and a physical model and, he developed a model of salamander amphibian locomotion (Ijspeert, 2001). The most interesting feature of this robot is modular heterogeneity. The body is a chain of homogeneous modules, and the robot also has four rotating limbs. While in the water, the robot should perform a swimming activity, mainly through undulation of its body. On the ground, the limbs are coordinated among themselves and with the body, which provides for maximum stride (Crespi et al., 2005; Ijspeert et al., 2005).

After the initial generation, Ijspeert proposed to use his robot to help test biological hypotheses. In (Ijspeert et al., 2007), they propose a plausible connectivity topology derived from research on real salamanders (figure 4.2b).

Besides Amphibot, the EPFL developed *Yet Another Modular Robot (Ya-*

[4] `http://www.epfl.ch/index.en.html`

[5] `http://biorob.epfl.ch/amphibot`

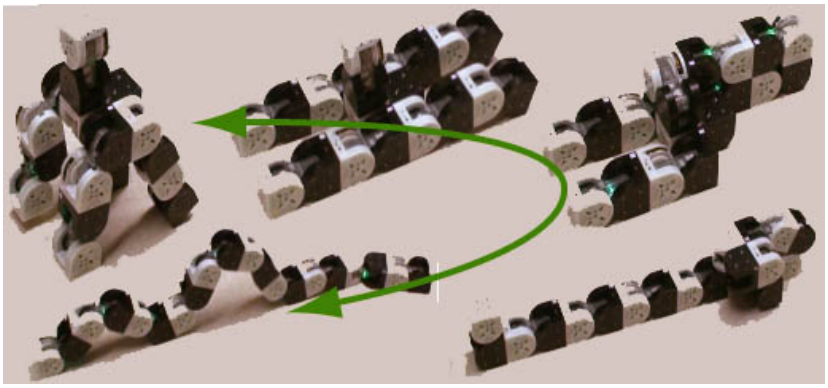


Figure 4.3: M-TRAN III. Photograph courtesy of H. Kurokawa, AIST

MoR). This platform is perhaps less bio-mimetic and more powerful, technically speaking. Special attention was put onto developing a good reconfiguration mechanism as well as equipping modules with bluetooth connectivity and other features. The most important contributions made on this platform approach the issue of on-line gait learning. The controller is a very simple chain of coupled non-linear oscillators. An on-line algorithm is run to determine speed of travel (Marbach and Ijspeert, 2005) and to find new locomotion gaits (Sproewitz et al., 2008).

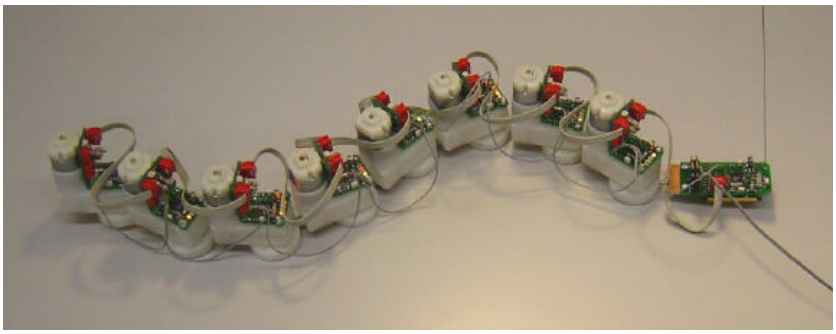
4.4 National Institute of Advanced Industrial Science and Technology

The *National Institute of Advanced Industrial Science and Technology (AIST)*[6] is located in Tsukuba and Tokyo, Japan

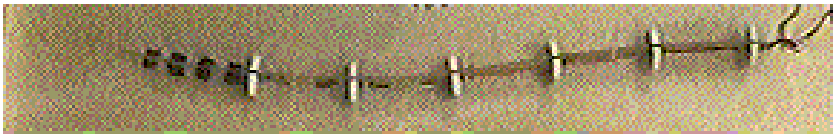
The most representative example of modular robots in the AIST is the *Modular Transformer (M-TRAN)* family, comprising three generations of robots (M-TRAN III in figure 4.3). M-TRAN's goal is online modular reconfigurability (Murata et al., 2002). Following works introduced the notion of neural control to different degrees (Kamimura et al., 2003, 2004). These robots pose a still open control question, namely *how to reconfigure locomotion when the physical structure of the robot is reconfigured.*

The solution found by Kamimura et al. is offline CPG reconfiguration. The authors foresee a small number of possible configurations, and using a genetic algorithm they search for a suitable neural topology. Each module is controlled by a Matsuoka non-linear oscillator (Matsuoka, 1987). Then, the search algorithm adjusts a connectivity matrix between all oscillators of the

[6] `http://www.aist.go.jp`



(a) Wormbot



(b) Robotic lamprey

Figure 4.4: Photographs courtesy of (a) Jörg Conradt, ETHZ, (b) Joseph Ayers, NEU

robot and simulates the resulting CPG and robot in a virtual environment.

4.5 Miscellanea

There are other works in the area that deserve attention. Previous examples are families of robots that have had great impact on the community. The following are two examples of interesting modular robots whose development has unfortunately been discontinued.

The first CPG controlled lamprey robot (figure 4.4b) that the author has knowledge of was developed at the Marine Science Center, Northeastern University (NEU), USA[7] (Ayers et al., 2000). This robot is the result from a systematic study of real lamprey locomotion kinematics. It combines control aspects including inter-module coordination mechanisms with a novel nitinol based actuator system. These actuators resemble actual muscles of living lampreys, which makes this model suitable for testing hypotheses about the actual mechanisms of locomotion generation and coordination.

Shortly after the lamprey robot, the Institute of Neuroinformatics at the Eitgenössische Technische Hochschüle Zurich (ETHZ) developed *WormBot*[8] (figure 4.4a) (Conradt and Varshavskaya, 2003). Each module is controlled by a phase oscillator with explicit frequency control and sinusoidal coupling

[7] `http://www.neurotechnology.neu.edu/lamprey.html`

[8] `http://www.ini.uzh.ch/~conradt/projects/WormBot`

to first neighbors, a model adapted from (Cohen et al., 1982). This is a very simplified model of locomotion coordination, limited to oscillatory activity. We believe that recent research on CPGs will open up new possibilities for richer, more complex CPG robotic control.

5 Dynamical Models

It would be wise to gain a basic understanding of the formalism of dynamical systems in order to understand the models explained hereafter in this thesis. In this section the reader will find a basic introduction to the concept of dynamical system and then some other important notions such as bifurcations and the difference between continuous and discrete time dynamical systems.

5.1 Dynamical systems

Every phenomenon is subject to the passage of time. All phenomena arise, have a duration and cease to exist. The description of how individual systems change over time is the subject of study of dynamical systems theory.

In order to describe the dynamics of a given system, we create a mathematical *model* that incorporates only relevant information about the system, and intentionally leaves out irrelevant or unknown facts. The specific assumptions of particular details such as the set of possible configurations of the system or how time develops will determine the formalism under which the model will be defined. In this thesis we will only consider systems of *Ordinary Differential Equations* (ODE) and *Discrete Dynamical Systems* (DDS), and will intentionally leave out other common modeling techniques such as cellular automata or Petri nets.

5.1.1 Continuous and discrete dynamical systems

When building a dynamical description of a system, each one of the features considered relevant should be described through a variable in the model. The next step is to describe the evolution and possible interaction between these relevant features. If we consider time as a continuous, real valued magnitude, a system may be modeled using ODEs as:

$$\dot{\vec{x}} = F(\vec{x}, \vec{p}) \tag{5.1}$$

where $\vec{x}$ is the set of relevant variables, F is the set of functions representing the dynamics of the system, and $\vec{p}$ is the set of parameters shaping function F. The set over which $\vec{x}$ takes its values is called the *phase space* of the system,

which encompasses all the possible configurations thereof. In order to predict the state of the system at any given time, with given initial conditions, one needs to integrate F over time.

However, given that oftentimes dynamical systems are simulated on a computer, it may be more appropriate to treat time as a discrete magnitude. In such cases, a dynamical system may be modeled as a DDS:

$$\vec{x}_{n+1} = f(\vec{x}_n, \vec{p}) \tag{5.2}$$

where $\vec{x}$, F and $\vec{p}$ have the same meaning as in 5.1, and time is accounted for through variable n.

5.1.2 Understanding discrete time dynamical systems

As any other dynamical system, discrete dynamical systems may display a wide range of behaviors, depending on the topology of its equations and the combination of parameters. In general, a discrete time dynamical system is a mapping from $\mathbb{R}^n$ into $\mathbb{R}^n$, parametrized by a vector of parameters $\vec{p}$:

$$\vec{x}_{n+1} = f(\vec{x}_n, \vec{p}) \tag{5.3}$$

It is important to note that this system is discrete in time, but its state space can be continuous.

A geometrical tool: the return map

A useful tool in understanding discrete dynamical systems is the return map, which plots x_{n+m} against x_n, with $m = 1$ usually. This tool is restricted to one-dimensional systems, but it will nevertheless help us in our study. The main contribution of this tool is that we can visualize how the system evolves with time from a particular initial condition. The procedure is easy, take for instance figure 5.1, then:

1. Start on the abscissæ and find your initial condition x_0.

2. Trace a vertical line that meets the point $(x_0, f(x_0))$. This point reveals the state of the system in the next step, since $x_1 = f(x_0)$.

3. Trace a horizontal line that meets the identity diagonal. This point is (x_1, x_1).

4. Repeat steps 1-3 for subsequent points.

Let us consider the discrete dynamical system of one dimension defined by:

$$x_{n+1} = A(x_n^3 - c^2 x_n) \tag{5.4}$$

A system governed by this simple equation will have a well-defined, deterministic temporal evolution, depending on its initial conditions and the values of its parameters. To study the possible set of behaviors of the system, we can give A and c concrete values and try different examples. For instance, $A = 3$, $c = 0.75$, and then $A = 6$ and $c = 0.75$ (see figure 5.1). There is a clear qualitative change in the way the system behaves. In the first case, initial conditions within a given interval always converge to a stable oscillation of period 2. In the second case, all trajectories diverge to infinity. This shows that, even though the *structure* of the equations is the same, the actual values of the parameters can make a qualitative difference on the behavior exhibited.

The only different between the first and the second examples is the value of parameter A. Clearly, the qualitative behavior of the system will depend on this parameter, so it is worth studying what different types of behavior can result from different values.

Parameter analysis: bifurcations

The return plot is a great tool to visualize the possible trajectories of the system, but it can also reveal *structure* in it, as well as points of interest. In figure 5.1, several points of interest can be seen: the points where the curve meets the identity, the zero crossing and the two peaks of the curve. In this figure, the peaks of the curve, in absolute value, are below the crossings of the curve with the identity. What if we would change parameter A so that the peaks are above the diagonal crossings?

This is the difference between panels in figure 5.1. With a simple change of parameter A, the system behaves qualitatively very differently. This change in the qualitative behavior of a system depending on the value of its parameters is called a *bifurcation*.

Figure 5.2 is a so called *bifurcation diagram*, and shows the convergent behavior of the system for different values of parameter A. To build it, we set different values of A and let the system evolve a very long time from arbitrary initial conditions, long enough for it to achieve a stable regime. Then, we plot on a vertical line over the corresponding value of A the different values that x_n takes. If the system is in an equilibrium state ($x_{n+1} = x_n$), there will only be one point for that value of A. If the system oscillates with a rhythmic pattern of period two, there will be two points, and so on.

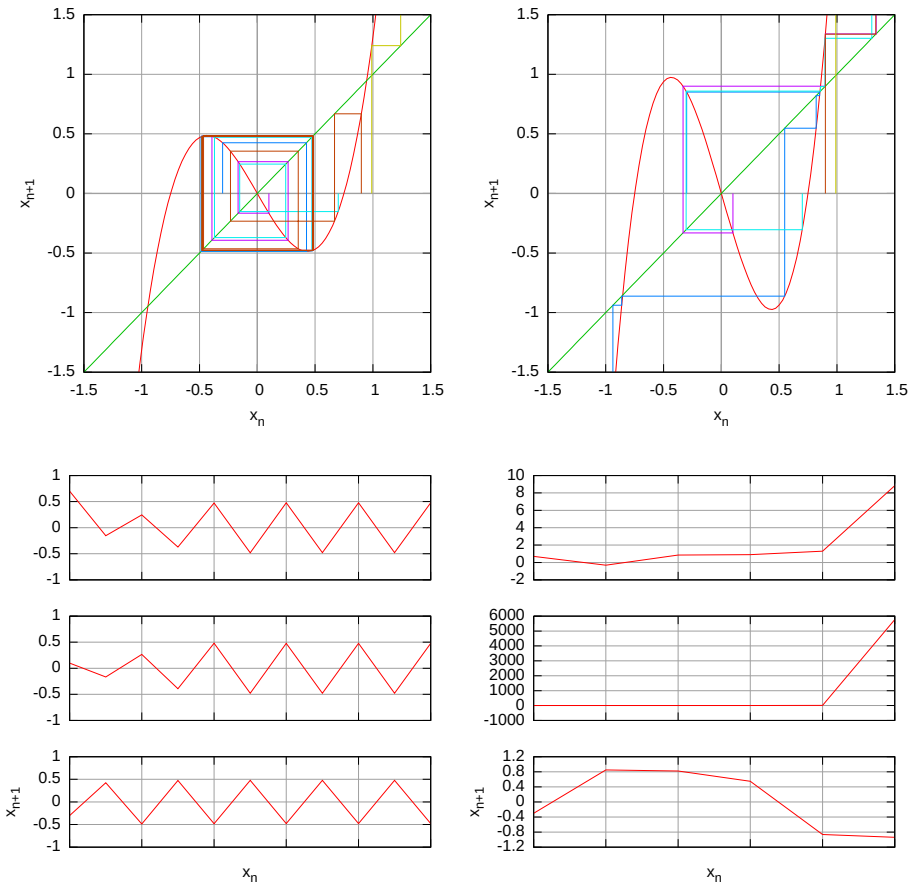


Figure 5.1: Return maps of the discrete dynamical system $x_{n+1} = A(x_n^3 - 0.75^2 x_n)$ for different initial conditions. Left panels show stable orbits for $A = 3$ while right panels show unstable orbits at $A = 6$. Despite different initial conditions, all stable trajectories converge to an oscillatory orbit of period two: {-0.5, 0.5}. Unstable orbit trajectories always diverge to infinity. A change in the values of the parameters of the system results in qualitatively different behaviors. Such a change is called a *bifurcation*. Upper panels: trajectories for initial conditions {-0.3, 0.1, 0.7, 0.9, 0.99}. Lower panels: time series for initial conditions {-0.3, 0.1, 0.7}.

There are clearly distinct qualitative behaviors for different intervals of A (see figure 5.3). For instance, for $A \in (0, 1.5)$, the system always converges to 0. In the interval $(1.5,\ 3.9)$, the system converges to a stable periodic orbit of period two, with amplitude depending, again, on the value of A. As the value of A increases, the system undergoes several transitions. The first one is a *period doubling* transition, which means that the stable periodic orbit shown by the system is of period four, instead of two. This transition is called a *bifurcation*. Interestingly, after some of these period doubling bifurcations the system begins to show chaotic behaviors. This means that the system is kept in a bounded region of its phase space, but there is no apparent pattern in its trajectory. This bifurcation diagram is similar to other chaotic systems, like the logistic map.

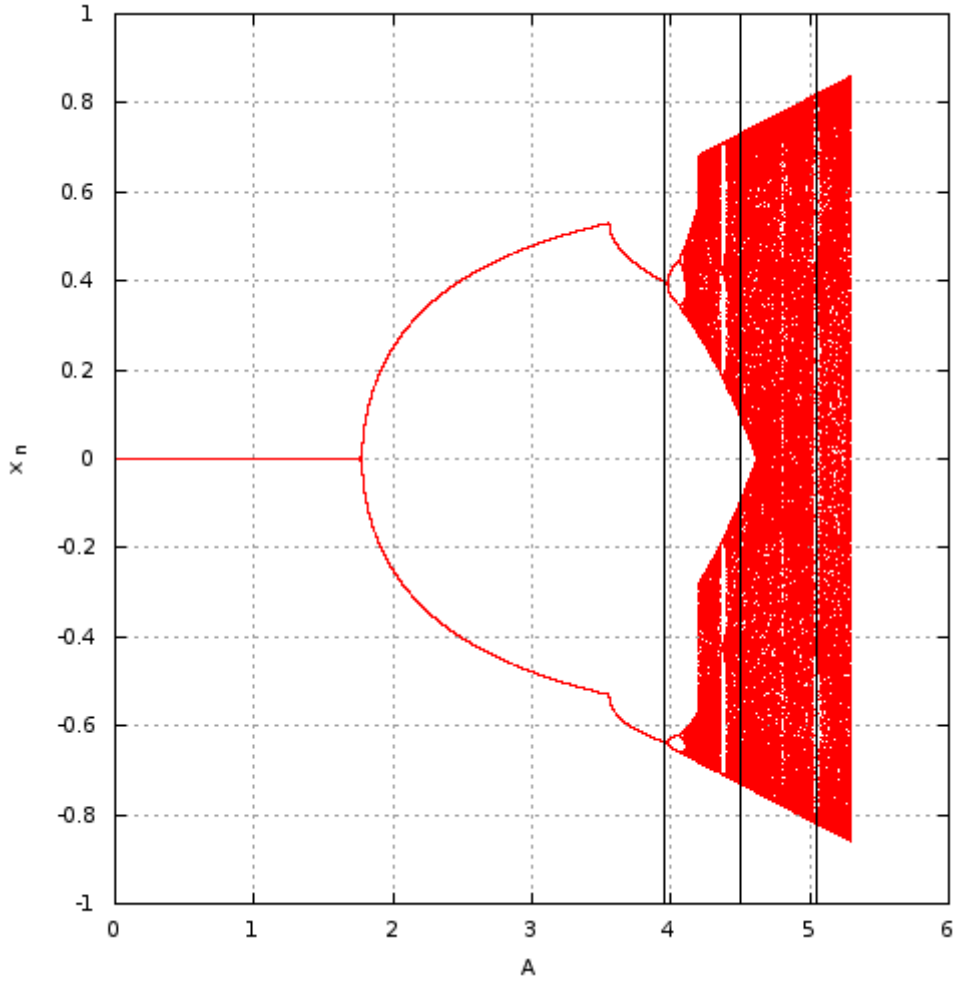


Figure 5.2: Bifurcation diagram of system $x_{n+1} = A(x_n^3 - c^2 x_n)$ with $c = 0.75$ and $x_0 = 0.5$. For each value A, the values to which the system converges are plotted. For higher values of A the system diverges to infinity. Interestingly, different qualitative regions can be observed. First, in the approximate interval (0, 1.5), the system always converges to zero. In the interval (1.5,3.9), the system converges to a stable periodic orbit of period 2. From there, the system undergoes several transitions called *period doubling* as A is increased. In some regions, the system displays chaotic behavior. Trajectories corresponding to designated vertical lines are shown in figure 5.3.

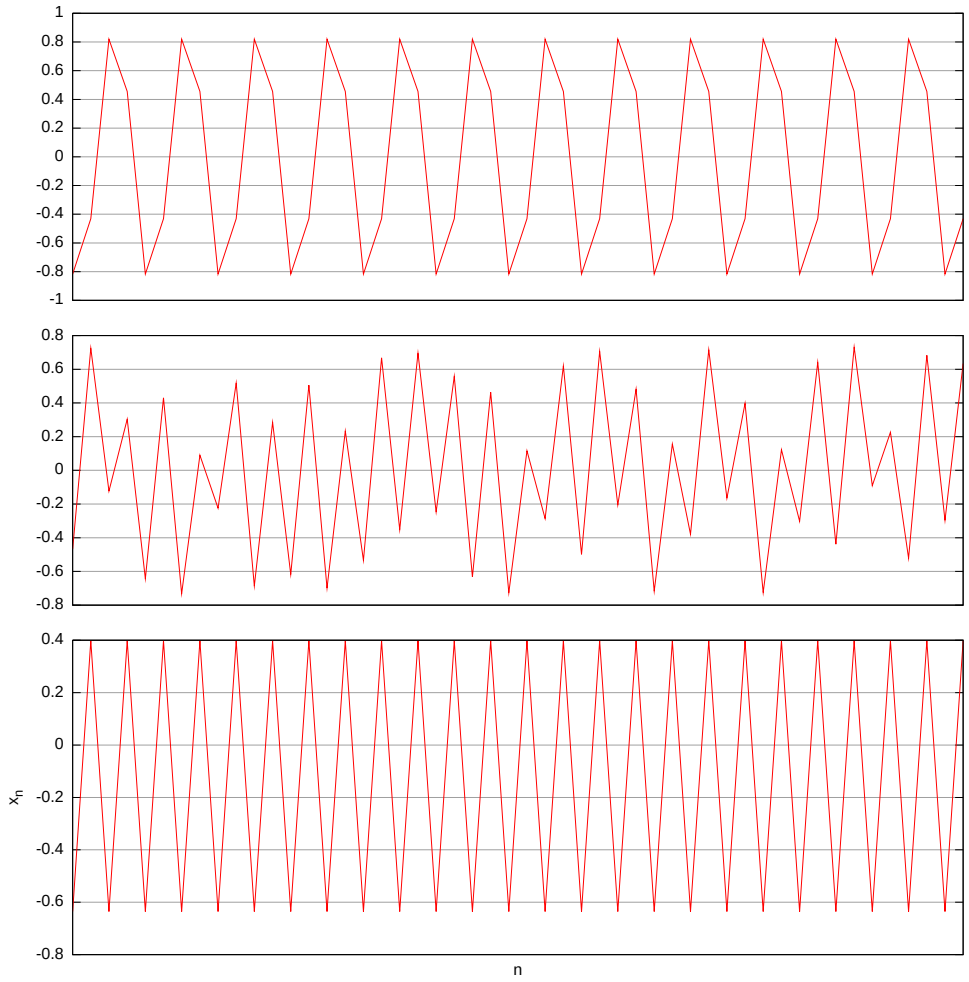


Figure 5.3: Time series for different values of A in the system $x_{n+1} = A(x_n^3 - 0.75^2 x_n)$ with initial conditions $x_0 = 0.5$. A dynamical system governed by a single equation can behave differently depending on its parameters. Different values of parameter A give a different structure to the phase space of the system. In this particular example, chaotic regions and periodic regions can be found in a wide range of values of A. Lower panel: A=3.9525, period two periodic orbit. Center panel: A=4.5055, chaotic trajectory. Top panel: A=5.05, period four periodic orbit. These simulations correspond to highlighted trajectories in figure 5.2

5.2 Bio-inspired CPG components

We want to build CPGs that meet the requirements of autonomy, flexibility, robustness and controllability. There are several apparent contradictions here. One of the main benefits of employing CPGs as locomotion controllers is that higher-level control centers can operate with lower dimensional control systems. The responsibility of, for example, planning goals and controlling joints is then decoupled, so that the complexity of coordinating limbs is managed by a CPG, and the planning center will only issue commands like "activate" or "deactivate". For that reason is autonomy desirable in a CPG.

However, autonomy is precisely the quality of not being externally controlled. CPGs manage this duality very well. While being able to generate a robust rhythm without intervention from outside entities, they also incorporate external commands that can modulate frequency or amplitude, for example.

Similarly robustness and flexibility are a seeming contradiction. We want CPGs to generate stable and reproducible rhythms, but being able to *configure* the widest possible range of rhythms is also desirable. CPGs are able to generate very precise rhythms, but they are also able to change their working regime according to circumstances.

To achieve all of this, we look for dynamical elements with the appropriate intrinsic dynamics, robust enough to produce stable rhythms, but flexible, with smooth transient behavior to negotiate with other elements and to adapt to external conditions.

5.2.1 A multiple time-scales neuron model

The neuron model developed by Rulkov et al. (Rulkov, 2002) presents key dynamic properties of living neurons: multiple time scales and different working regimes. It is mathematically simple and the possible set of behaviors can be controlled depending on the selection of parameters. Three stable regimes may be selected by combination of its parameters: silent, in which the potential of the neuron (variable x_n in eq. (5.6)) remains in a constant resting state; tonic spiking, in which the neuron emits spikes at a constant rate; and tonic bursting, in which bursts of spikes are emitted at a constant rate, with a silent interval in between. Furthermore, in the boundaries of the parametric regions of those regimes, chaotic behavior may be found (Shilnikov and Rulkov, 2003). Of these behaviors, tonic bursting is the one of greater interest for us. See Figure 5.4 for an overall idea of the model working in tonic bursting regime. These regimes, both of silence and intrinsic oscillations are observed in living CPG neurons (Selverston et al., 2000).

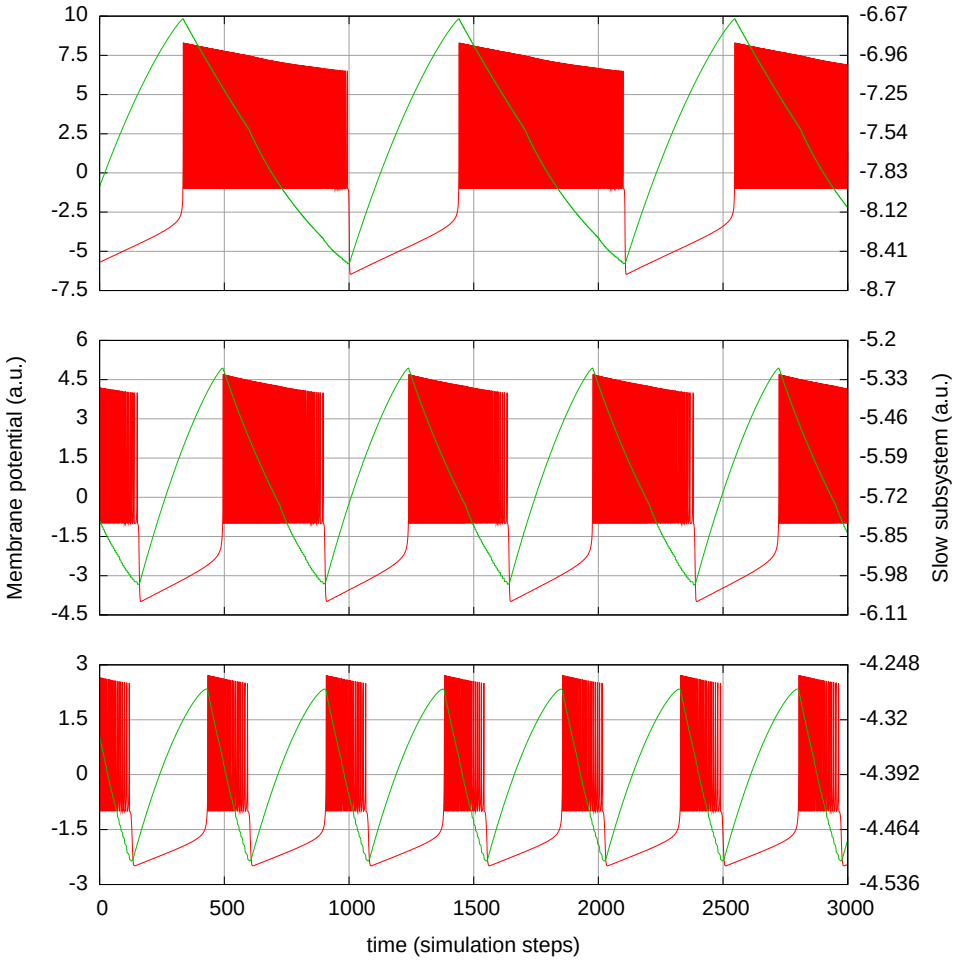


Figure 5.4: Fast and slow subsystems of Rulkov's neuron model. The slow subsystem (y_n in (5.7)) is responsible for signaling the beginning and end of a burst; the fast subsystem (x_n in (5.6)) is responsible for the oscillations that generate individual spikes within each burst. Bottom panel: $\alpha = 7$ and $\sigma = -0.33$; Center panel: $\alpha = 10$ and $\sigma = 0$; Top panel: $\alpha = 15$ and $\sigma = 0.33$.

The mathematical description of Rulkov's model as used in this work is as follows:

$$f(x,y) = \begin{cases} \frac{\alpha}{1-x} + y, & \text{if } x \leq 0 \\ \alpha + y, & \text{if } 0 \leq x < \alpha + y \\ -1, & \text{otherwise} \end{cases} \quad (5.5)$$

$$x_{n+1} = f(x_n, y_n + \beta_e I_n) \quad (5.6)$$

$$y_{n+1} = y_n - \mu(x_n + 1) + \mu\sigma + \mu\sigma_e I_n \quad (5.7)$$

with $\mu = 0.001$ in all experiments.

This is a bi-dimensional model, where variable x_n represents a neuron's membrane voltage and y_n is a slow dynamics variable with no direct biological meaning, but with similar meaning as *gating* variables in biological models that represent the fraction of open ion-channels in the cell. While x_n oscillates on a fast time scale, representing individual spikes of the neuron, y_n keeps track of the bursting cycle, a sort of context memory. Units are dimensionless, that is, one can rescale them to match the requirements of the robot.

The combination of σ and α selects the working regime of the model: silent, tonic spiking or tonic bursting. In the bursting regime, these parameters also control several properties of neural activity, as shown in Figure 5.5. It is through these parameters that we can tune the characteristics of the final locomotion.

Finally, external input is modeled through I_n. This property is essential for autonomous organization: processing units in the CPG must be able to negotiate the rhythm among them. Also, entrainment between the CPG and the physical robot can be achieved through I_n by adding an error term as external input to a neuron. The total effect of this parameter will depend upon past history of events, the exact value of I_n and the phase within the burst cycle at which the neuron finds itself. Parameters σ_e and β_e will control how external currents affect the fast and the slow subsystem.

Effect of parameters on the neuron model

From a control perspective, it is interesting to know the behavior of a neuron as a function of its parameters. Figure 5.5 shows the relationship between parameters α and σ and different properties of neural activity: period, duty cycle and number of spikes per burst. There is a clear, quasi-linear relationship between bursting period and parameter α, while the influence of σ is almost irrelevant in this respect. On the other hand, the relationship between bursting time and total period is mostly dependent on σ. With only these two parameters the neuron model can be configured in a wide range of configurations.

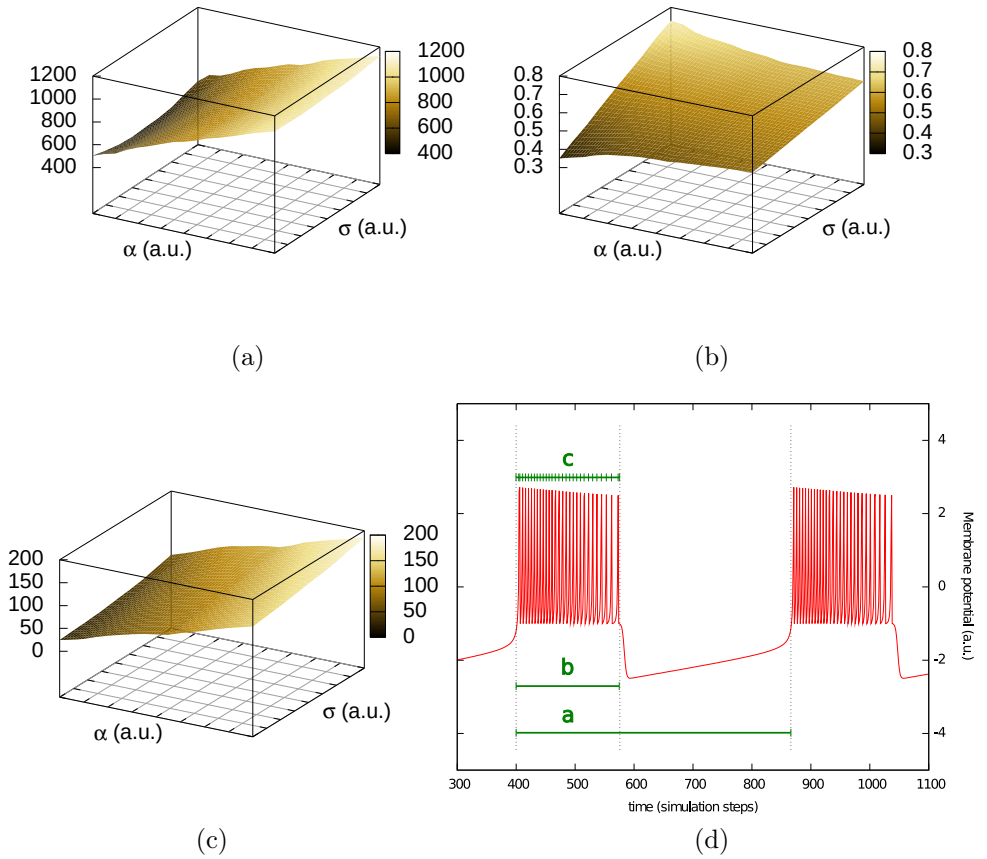


Figure 5.5: Different properties of an isolated neuron (Rulkov's model) in bursting regime for different values of α (from 6 to 16) and σ (-5 to 5). Thirty consecutive bursts in stable regime were analyzed: (a) Mean period, measured in simulation steps; (b) Mean duty cycle, measured as the percentage of the period that corresponds to spiking activity; (c) Mean number of spikes per burst of thirty consecutive bursts. (d) Explanation of magnitudes: a is period, b/a is duty cycle and c is the number of spikes per burst.

5.2.2 Synapse model

A key property of CPGs is that they are autonomous, i.e. the different units in the circuit talk to each other to negotiate the overall function. Here we present the model we have chosen to implement synapses, the communication channel of neurons. In this work we use a synapse model by Destexhe et al. (Destexhe et al., 1994). It models a chemical synapse that, upon arrival of an action potential (a spike of the neuron), releases a neurotransmitter during a certain amount of time.

Chemical synapses are unidirectional. We call *presynaptic* the neuron "before" the synapse and *postsynaptic* the one "after" it . When a potential spike arrives from the presynaptic neuron, the synapse releases a certain amount of neurotransmitter molecules that bind to the postsynaptic neuron's receptors. With time, neurotransmitter molecules start to "unbind". If a succession of spikes arrives within a short time, the synaptic response to each of them may overlap. Therefore the state of the synapse is dependent upon past events, a kind of context memory (see figure 5.6).

The additional time-scale provided by kinetic synapses in a CPG enriches synchronization between bursting neurons. For instance, we may choose to synchronize two bursting neurons upon the spike (fast) time scale or the burst (slow) time scale. We have selected the kinetics of the binding and unbinding processes such that synapses act as filters of the fast time scale and synchronization occurs at the slow time scale. That is, the basic unit of synchronization will be the burst as a whole, not every individual spike. Beyond this, synapses may introduce delays for richer control of phase difference between neurons.

The mathematical description of the model follows:

$$\dot{r} = \begin{cases} [T]\lambda(1-r) - \beta r, & \text{if } t_f < t < t_f + t_r \\ -\beta r, & otherwise \end{cases}$$

This equation defines the ratio of bound chemical receptors in the postsynaptic neuron (see Figure 5.6 for a sample trace), where r is the fraction of bound receptors, λ and β are the forward and backward rate constants for transmitter binding and $[T]$ is neurotransmitter concentration. The equation is defined piecewise, depending on the specific times when the presynaptic neuron fires (t_f): during t_r units of time, the synapse is considered to be releasing neurotransmitters that bind to the postsynaptic neuron. After the release period, no more neurotransmitter is released and the only active process is that of unbinding, as described by the second part of the equation. Times t_f are determined as the times when the presynaptic neuron's membrane potential crosses a given threshold θ.

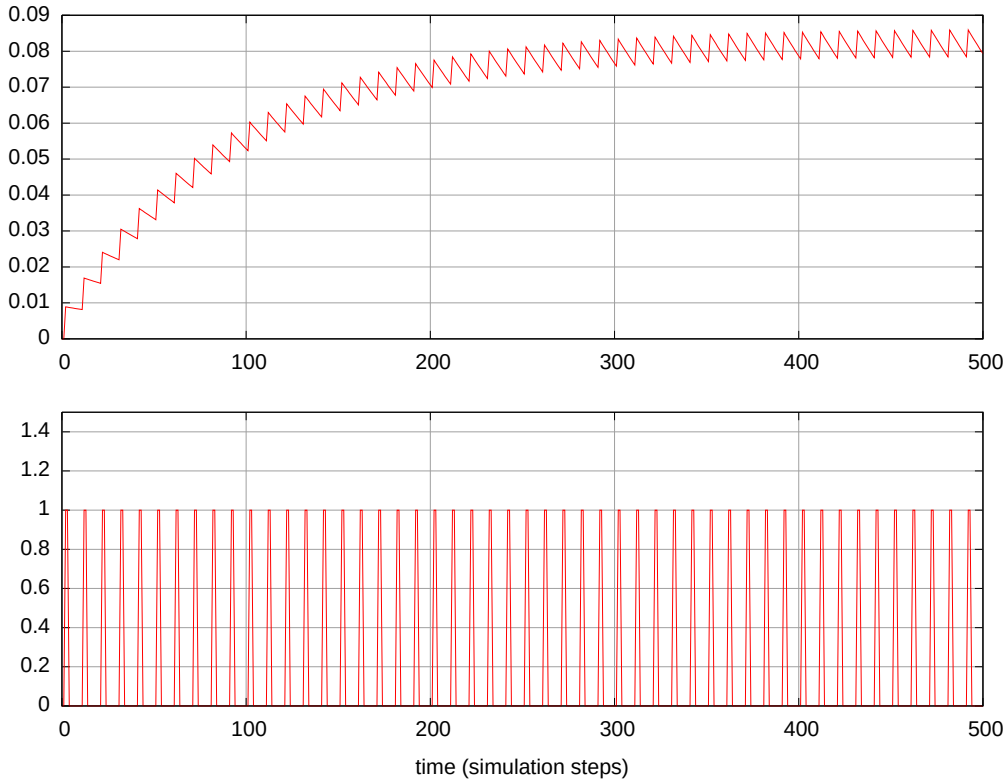


Figure 5.6: Synaptic response (upper panel) to a train of spikes arriving from the presynaptic neuron (lower panel). For each spike from the presynaptic neuron, a small amount of neurotransmitter is released in the synapse, that binds to receptors in the postsynaptic neuron. After a short time, transmitters begin to unbind from receptors. The process of binding and unbinding cause the characteristic sawtooth shape. This model of synapse shows a memory effect, in the sense that response to any given pulse depends on past history of events. Variable r of the model (upper panel) represents the rate of bound receptors in the postsynaptic neuron. According to equation (5.8), the current that will flow into the postsynaptic neuron is proportional to the rate of bound receptors. That is, when a neuron emits a series of spikes, the current that will flow into the postsynaptic neuron has a dynamic behavior, rising with time and converging to a stable value.

Synaptic current is then calculated as follows:

$$I(t) = g \cdot r(t) \cdot (X_{post}(t) - E_{syn}) \tag{5.8}$$

where $I(t)$ is postsynaptic current at time t, g is synaptic conductance, $r(t)$ is the fraction of bound receptors at time t, $X_{post}(t)$ is the postsynaptic neuron's membrane potential and E_{syn} its reversal potential, the potential at which the net ionic flow through the membrane is zero. When coupling two Rulkov map neurons we will need to use a discrete synaptic function. We will build a sequence, let us call it I_n, by simulating $I(t)$ as a continuous function and then taking samples every 0.01 time units (for our choice of kinetic parameters as outlined in the different figures).

We say that a synapse is excitatory when the probability of the postsynaptic neuron firing a spike increases after the presynaptic neuron has fired. If the probability decreases, the synapse is inhibitory. If the postsynaptic neuron rhythmically emits spikes, an excitatory synapse will generally increase its frequency while an inhibitory one will generally decrease it.

5.3 Analysis tools

When evaluating a particular CPG model, we are interested in knowing if the resulting locomotion gait is effective and its parameters are within reasonable ranges. In particular, the CPGs designed in this thesis will be generating symmetric oscillations, so we will be most interested in verifying their amplitude, frequency, and phase difference between modules.

In order to analyze one signal generated for one motor by the CPGs, we first generate an analytic signal that uniquely represents it. The basic tool for our analysis is the Hilbert transform[1]. Basically, the Hilbert transform of a real-valued function/sequence is another real-valued function/sequence whose Fourier components are shifted 90º with respect to the original.

Let $m = \{m_1, m_2, \ldots, m_n\}$ be a signal generated for one of the motors (thus a real-valued sequence). We will denote $\mathcal{H}\{m\}$ the Hilbert transform of the sequence, which is also real-valued and $\mathcal{H}\{m\}_i$ the i-th element of the resulting transformed sequence. The equivalent analytic signal representation of m is:

$$Z = \{Z_i = m_i + j \cdot \mathcal{H}\{m\}_i\} \tag{5.9}$$

[1] a complete revision of the theory can be found in (Oppenheim et al., 1999)

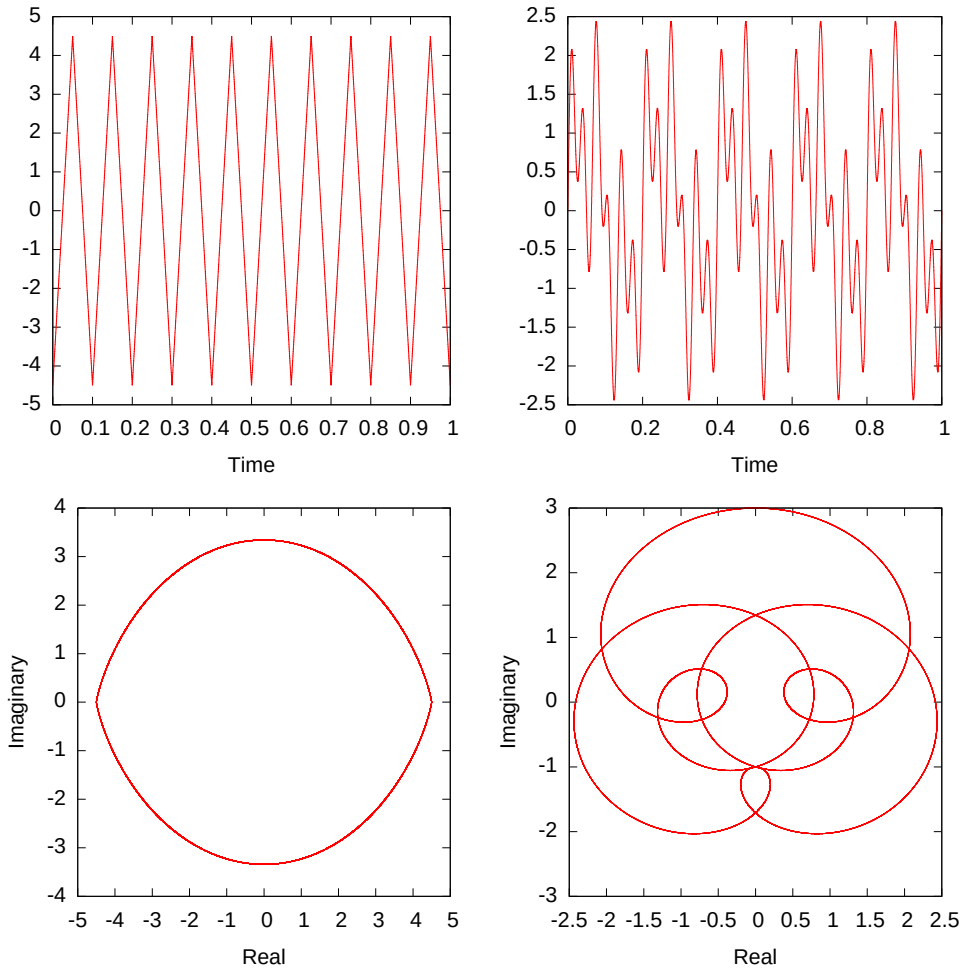


Figure 5.7: Sample analytic signals from real-valued signals. The first two examples are perfectly periodic mono-component signals with different shapes. The structure of the projected analytic signal is that of an orbit centered at the origin, with shape reflecting the shape of the oscillations. Phase of one point is the angle of that point in polar coordinates. Even though instantaneous phase can always be mathematically defined, it does not always carry a physical meaning. In the last two examples, there are different points within one single period with equal phase value due to the loops. Upper row: (a) $\sin(2\pi 5t)$ (b) $\sin(2\pi 5t) + \sin(2\pi 13t)$. Lower row: complex plane projections along the time axis of the corresponding analytic signals of the upper row.

This representation allows us to model Z as an amplitude and phase modulated oscillator

$$Z_i = A_i e^{j\phi_i} \tag{5.10}$$

such that

$$A_i = |Z_i| = \sqrt{m_i^2 + \mathcal{H}\{m\}_i^2} \tag{5.11}$$

and

$$\phi_i = \angle Z_i = \tan^{-1}\left(\frac{\mathcal{H}\{m\}_i}{m_i}\right) \tag{5.12}$$

Furthermore, we can now reconstruct the original signal m as:

$$m_i = A_i \cos(\phi_i) \tag{5.13}$$

There has been extensive discussion about the applicability of this technique in general cases, particularly when calculating the instantaneous frequency derived from instantaneous phase (Huang et al., 1998). The two main restrictions applicable to signals for their instantaneous frequencies to make physical sense are that they be mono-component and that they oscillate symmetrically around zero. Loosely speaking, a mono-component signal is one which does not have sub-oscillations between zero-crossings, i.e., there is only one local extreme between zero-crossings.

In order to gain some insight as to how the analytic signal is related to the original signal, we present two examples in Figure 5.7. In this figure we show two signals and the complex plane projection of their corresponding analytic signals. The first two signals are mono-component signals. Their respective analytic signals projected on the complex plane show an oscillatory orbit around the origin. Phase calculated as in (5.12) will yield a monotonically increasing value within each period of the signal. The other two cases are instances of non-monocomponent signals. In these cases, since the projected signal does not describe a simple orbit but displays some loops, there will be at least two distinct points within one period that will have equal phase value.

Signals of our CPGs have been filtered before plotting with a moving average filter. The window size is 1000 points of width, a little less than the mean period of the signals. This way we eliminate small noise in the plot and keep the general behavior. Noise in the original signal does not result in jerky locomotion of the robot as can be seen in the on-line videos.

5.3.1 Phase difference

Efficient locomotion of the robot is achieved when adjacent modules maintain a constant phase difference between them in the steady state. In order to study phase differences, we take signals $m^{(k)}$ and $m^{(l)}, k \neq l$ corresponding to motors k and l and construct $Z^{(k)}$ and $Z^{(l)}$. From these, we calculate phase difference between $m^{(k)}$ and $m^{(l)}$ making use of (5.12) as:

$$\Phi^{(kl)} = \{\Phi_i^{(kl)} = \phi_i^{(k)} - \phi_i^{(l)} = angle(Z_i^{(k)} \cdot Z_i^{(l)*})\} \tag{5.14}$$

where product with the complex conjugate is used to subtract the angle of $Z_i^{(l)}$ from the angle of $Z_i^{(k)}$.

5.3.2 Frequency

Having defined the instantaneous phase of a signal in (5.12), we now define the instantaneous frequency of a signal as:

$$\omega_i = \phi_i - \phi_{i-1} \tag{5.15}$$

Note that this is an instantaneous value of frequency, and considerations in (Boashash, 1992) apply here when interpreting the meaning of this value.

Part II

Results

6 Modular Oscillator

A neuron and a synapse model have been introduced in the previous chapter. The neuron model 5.7 is a simple discrete map with rich intrinsic dynamics, suitable for rhythm negotiation. Depending on parameter configuration, it may reproduce silent, tonic spiking, tonic bursting or chaotic bursting dynamics. The synapse model 5.8 reproduces the dynamics of chemical synapses. It is a one-dimensional model whose temporal dynamics can be configured with two time constants. We use the dynamical features of these two models to implement efficient autonomous rhythm negotiation strategies.

In this thesis we follow a bottom-up approach. First, a modular controller for a single robot module will be introduced. We will study its control properties, and in particular we will show that the rhythm produced is predictable and robust in the presence of noise. Then, we will show that this modular oscillator is at the same time very flexible and can adapt its rhythm according to the response of the servo. Finally, we will devise different negotiation topologies, based on different principles to interconnect the modular oscillators so that they will autonomously produce effective locomotion.

6.1 Half-center oscillator

Robots are articulated machines. Robotic control often involves the development of a kinematic model in which the position of joints with different degrees of freedom must be taken into account.

One degree of freedom (1DOF) joints are very versatile despite their apparently reduced flexibility. By a judicious combination of one degree of freedom joints and higher dimensional joints, complex trajectories may be performed by a robot. Take as an example a human arm: the shoulder has a high number of degrees of freedom; following it is the wrist, with less degrees of freedom, only two; but the majority of joints have only one degree of freedom, including elbow and finger joints.

In nature, one degree of freedom joints are usually operated by two antagonistic groups of muscles. One group of muscles promotes motion in one direction while the other group promotes motion in the exactly opposite

direction. These groups receive the names of *promotor* and *remotor*, or *protractor* and *retractor* muscles. Biceps and triceps in the human arm are a good example of this.

For a correct operation of the joint, promotor and remotor muscle groups must act in consonance. If the joint is to be fixed at a given angle, both groups must exert a balanced force for it to compensate external forces. If the joint is to be rotated, one group will have to stop exerting force for the other to operate. Consequently, if the task in which the joint is involved requires rhythmic activity, there must be a device that guarantees alternating activation of one group and then the other group.

Let us consider the case in which one neuron is responsible for the activation of the promotor group and another neuron is responsible for the activation of the remotor group: every time the promotor neuron is active the promotor group will contract, and if the promotor neuron is resting the promotor muscle group will relax, and equivalently for the remotor counterpart. By coupling these two neurons with mutual inhibitory connections, anti-phase synchronization of the two antagonistic muscle groups can be achieved. That is, the activation of one neuron will activate its muscle group and force the opposite neuron to be silent through inhibition. When the active neuron reaches a *fatigue* state, it will become silent, releasing the other neuron from inhibition, who in turn will become active and inhibit the first one, and so on.

The fact that neurons cannot sustain bursts for an indefinite period of time, guarantees that the inhibited neuron will be released at some point. If the period between successive bursts of the inhibiting neuron is long enough, the inhibited one will have a chance to burst and inhibit the former neuron itself. The dynamics that arises in a group of neurons interconnected through inhibitory synapses with a release mechanism is called *winner-less competition* (Afraimovich et al., 2004).

The minimal implementation of such dynamics is the *half center oscillator*, illustrated in figure 6.1. It was first proposed in (Brown, 1911) and is found in many animal species. Different mathematical models of this principle have been proposed. The one by Matsuoka (Matsuoka, 1987) has seen wide acceptance and many works have used this oscillator as their basis for rhythmic activity generation in robots (see for instance (Kurokawa et al., 2008; Matsuo and Ishii, 2007; Liu et al., 2008)).

We propose an oscillator built with a similar architecture, with richer neuron and synapse dynamics. This will provide for a robust yet flexible controller whose properties we will study in the following sections. Two endogenously rhythmic neurons (R and P) are interconnected with inhibitory synapses. The role of inhibition is to prevent both neurons from firing at the

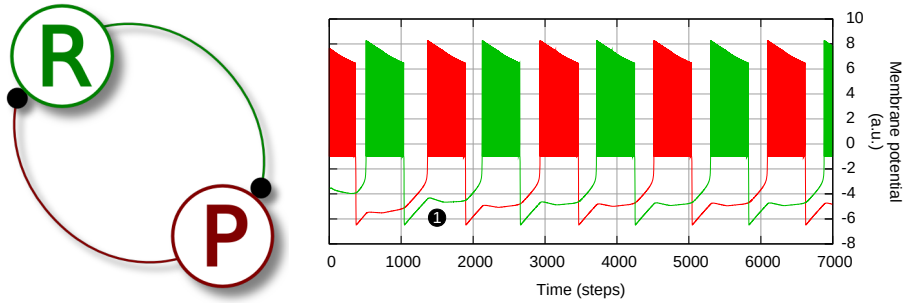


Figure 6.1: Half center oscillator. (Left) The promotor (P) and remotor (R) neurons are interconnected with strong inhibitory synapses. Each one of the neurons produces a rhythmic activity in the form of bursts of spikes. Whenever one of the neurons is bursting, the corresponding synapse will inhibit the other one. When the bursting neuron turns silent, the inhibited neuron will be released and will begin bursting, inhibiting the first one in its turn. (Right) Alternating activity sample of the promotor and the remotor neurons. Colors in the plot correspond to colors in the left diagram. While any one of the neurons is bursting, the other neuron is held at a low membrane potential (see label '1'). The combination of intrinsic neural and synaptic dynamics enables a self-coordinated, alternating bursting activity within the oscillator.

same time: when one fires, the other's activity is delayed; in turn, when the second neuron is bursting, it delays the first neuron's activity and the cycle begins again. It is worth noting that bursting neurons are flexible enough to negotiate a global rhythm while still being able to independently code locomotion information. Thus, after a process of synchronization, in which each neuron is capable of encoding their own information, they both arrive at a steady anti-phase state.

6.2 Translating from neural code to motor actuator commands: motoneurons

Bursting neurons, as used in this thesis, are rich dynamical systems in the sense that they contain two coupled subsystems: a slow and a fast one. Taking advantage of these multiple time scales we can separate the synchronization mechanism (slow neuron subsystem) from the locomotion encoding mechanism (fast neuron subsystem). Actually, even though they are mathematically coupled, they are relatively independent. Recalling figure 5.4, the slow subsystem is responsible for the activation and deactivation of the fast subsystem. The fast subsystem, in turn, is responsible for the generation of potential spikes. These spikes are translated to individual discrete communication events between neurons by chemical synapses. In the end, the actual message passed from one neuron to another depends both on the intrinsic slow and fast neuron dynamics, and synapse dynamics.

There is biological evidence that real bursting neurons have a reproducible spiking pattern (Szucs et al., 2003) that acts as a signature of the neuron. That is, just looking at the time distribution of spikes within one burst, one could recognize the specific pattern of a specific neuron. The converse is possibly also true, different spike distributions represent different messages, either from different neurons or in a different context.

Spike timing distribution is indeed a biologically plausible mechanism to encode information, and we use it to encode locomotion commands. In the half-center oscillator model, two neurons activate alternately to control one joint. When one neuron is active, it *pulls* the joint to one side. When it is inactive, no force should be performed on the joint on the respective side. Thus, the actual pattern of activation will shape the trajectory of the joint: each individual spike will trigger small *pulls* to one side.

A neuron called *motoneuron* is then responsible of *decoding* this information and translating it into the signal that will finally be sent to the actuator. In the case of the robot we study, this signal controls the angle of one servo. Figure 6.2 shows an example pattern of activity of an isolated module in its steady state, after an initial transient period of self-adjustment.

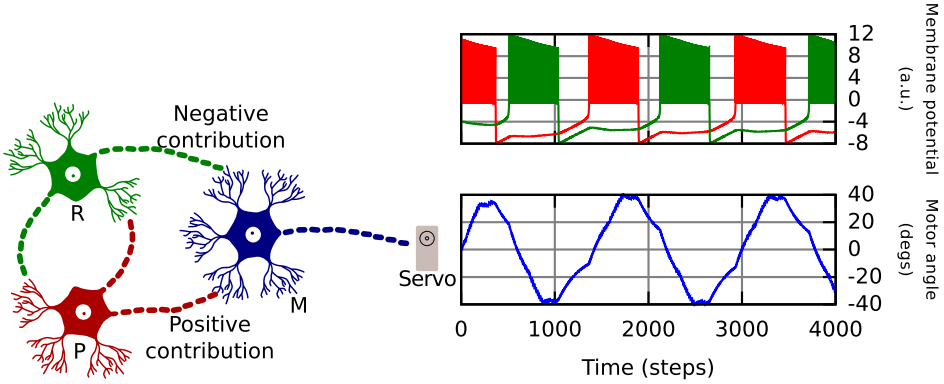


Figure 6.2: Activity sample of one module oscillator (variable x_n in (5.6)). (left) The promotor (P) and the remotor (R) neurons negotiate a common rhythm through inhibitory synapses. Mutual inhibition results in anti-phase synchronization. (right) The motoneuron (M) reads neural activity in the top panel and produces a control signal for the controlled joint, lower panel. Activation of the promotor neuron induces an increase of the output signal of the motoneuron, while activation of the remotor induces a decrease of the signal. The M neuron converges exponentially to the center value if both neurons are silent. Parameters for P and R: $\alpha = 15$, $\mu = 0.001$, $\sigma = -0.33$, $\beta_{\mathrm{e}} = 0$, $\sigma_{\mathrm{e}} = 1$; parameters for the inhibitory synapses between P and R: $\lambda = 0.5$, $\beta = 10$, $\tau = 0$, $E_{\mathrm{syn}} = 9$, $g_{\mathrm{syn}} = 1.5$, $T = 1$, $t_{\mathrm{r}} = 0.01$; Parameters for M: $\gamma = 60$, $\tau_{\mathrm{rise}} = 0.125$, $\tau_{\mathrm{fall}} = 0.25$, $\nu = -1.5$.

In (Thuma et al., 2003), biological evidence is provided that living muscles actually summate spikes of a burst (see also (Zhurov and Brezina, 2006; Brezina et al., 2000)). At the arrival of individual spikes from the motoneurons, living muscles suffer micro-contractions. If spikes arrive with a regular, sufficiently high frequency, muscles achieve a contraction state. This is what they call the *tonic component* of the neural command. When this state is achieved, muscles still show rhythmical micro-contractions provoked by each individual spike. This behavior is very similar to the motoneuron model that we introduce in this section.

Motoneurons *read* the activity of the half-center oscillator through a pair of synapses (see figure 6.2). These synapses connect the remotor and promotor neurons to the motoneuron, and are governed by a threshold equation:

$$s(x, \nu) = \begin{cases} 1, & \text{if } x > \nu \\ 0, & \text{otherwise} \end{cases} \tag{6.1}$$

The role of this function is to detect individual spikes of neurons. By setting the threshold to, for example, $\nu = -1.5$ *a.u.*, this function applied to the potential trace of one neuron will have value 1 during individual spikes and 0 otherwise.

The role of motoneuron M is now to integrate the individual events emitted by each one of the neurons. If neuron P emits a spike, motoneuron M will move the servo a little bit in a positive angle. If it emits a second spike close enough to the first one, the servo will be positioned a little bit further. Analogously, the R neuron will make the motoneuron move the servo towards negative angle positions. If both neurons are silent, motoneuron M will slowly drive the servo to a resting position of angle 0. This is accomplished through the following equation:

$$C(t) = \gamma[s(x_{\mathrm{p}}(t), \nu) - s(x_{\mathrm{r}}(t), \nu)] \tag{6.2}$$

$$\dot{m} = C(t)/\tau_{rise} - m(t)/\tau_{fall} + \text{offset} \tag{6.3}$$

where, $m(t)$ is the output of neuron M (in degrees), the two s terms are the threshold function (6.1) applied to input from R and P, τ_{rise} is a time constant that controls how quick the output signal $m(t)$ will rise, τ_{fall} controls how quick the output signal $m(t)$ will decline, and γ defines the maximum amplitude of signal $m(t)$. Parameter *offset* will add an offset so that the servo oscillates around that value instead of zero.

In this equation, P contributes positively and R negatively. Given the fact that P and R oscillate in anti-phase, the solution $m(t)$ is an oscillatory function bounded between $-\gamma$ and γ. When the motoneuron receives no

input because P and R are silent, it will go back to zero due to the leak term ($-m(t)$) in (6.3) (see decay between bursts in figure 6.2).

In the rest of this chapter we will study the properties of the oscillator. In particular, we will show the flexibility, adaptability and robustness of our design.

6.3 Properties of the oscillator

We will now explore the properties of the half-center oscillator that result from the combination of the intrinsic dynamics of neurons and synapses. First we will study the working range of the oscillator depending on different control parameters. Then we will show that it can autonomously adapt to a wide range of working conditions of the controlled joint, without any parameter change. Finally, we will study the robustness of the oscillator against noise, and compare its performance to another widely used oscillator model.

6.3.1 Flexible working range and influence of parameters

Self-organization is a key feature of CPGs. The half-center architecture shown in figure 6.1 has the ability to generate alternating bursting activity between the promotor and the remotor neuron without any external reference or control. The rhythm is negotiated among the two neurons through inhibitory synapses. The intrinsic dynamics of the neurons plays a central role in the overall pattern generation. We will see its effect on oscillation amplitude and period. Also the way in which each neuron incorporates synaptic activity into its own dynamics will shape the behavior of the oscillator. We will study three different possibilities: slow subsystem currents, fast subsystem currents and both slow and fast subsystem currents.

Rulkov's neuron model (eq. 5.6 and 5.7) has two coupled dynamical subsystems, a fast and a slow one, as discussed in section 5.2.1. Keeping all other parameters fixed, parameters α and σ select the working regime of the neuron between silent, tonic spiking and tonic bursting, and they also influence bursting period and burst length. The flexibility they provide directly translates to the oscillator built with this neuron model. Tuning α and σ a wide range of rhythmic patterns can be generated.

Besides this, two parameters in Rulkov's neuron model control how a neuron incorporates external input into its own dynamics: σ_e and β_e. Parameter σ_e modulates the effect of input currents on the slow subsystem of the neuron, and parameter β_e modulates their effect on the fast subsystem. Different combinations of these parameters will determine different working

dynamics for the oscillator.

Analysis will be carried out in three steps, according to different combinations of the last two parameters. For each combination, the role of α and σ will be studied. Ten independent simulations will be performed for the same combination of parameters, with different initial conditions. Amplitude and oscillation period will be calculated using the Hilbert transform as in section 5.3. The average amplitude and period is shown in the plots below.

Synaptic currents on the slow subsystem Setting $\sigma_e = 1$ and $\beta_e = 0$, currents flowing from one neuron to the other will have a direct effect on the slow subsystem of neurons, and only an indirect effect on the fast subsystem. Figure 6.3 shows the steady state amplitude and period of oscillation for this combination of parameters. The results are averaged over ten simulations for each pair of α and σ.

The resulting surface is very rough, indicating poor predictability of behavior. Such large variations in activity are surprising, and deserve further explanation. At closer inspection of the simulations, a seldom phenomenon is found to occur. Instead of the expected anti-phase synchronization of the slow subsystems of the promotor and remotor neurons that would yield steady oscillations, in-phase synchronization occurs between the slow subsystems and anti-phase between the fast subsystems, depending on initial conditions. Figure 6.4 illustrates this phenomenon.

We have studied the region of initial conditions where this phenomenon occurs. Figure 6.5 shows that the region where no oscillation happens is relatively small, but with probability greater than zero if neurons are initialized with arbitrary initial conditions. The figure shows mean amplitude and period of the motoneuron output depending on the initial conditions of the promotor and remotor neurons. To perform the experiments we first choose a reference point in the neuron's phase space. We choose a point on its limit cycle, corresponding to the onset of a burst. The promotor neuron's state is then set to this point, and advanced in time a given offset. The remotor neuron's state is set to the same state as the promotor neuron, and then advanced a different offset. With this experiment we try to measure the influence of the initial phase of neurons and the relative offset between them.

If we only take simulations with valid oscillations, the result is much more uniform and predictable, as shown in figure 6.6. There, a clear, almost linear relationship can be drawn with parameter α. Parameter σ has little influence, even though the relationship is also quasi-linear.

This configuration with $\sigma_e = 1$ and $\beta_e = 0$ is sensitive to initial conditions. Two equilibrium states can be achieved, one of which does not produce

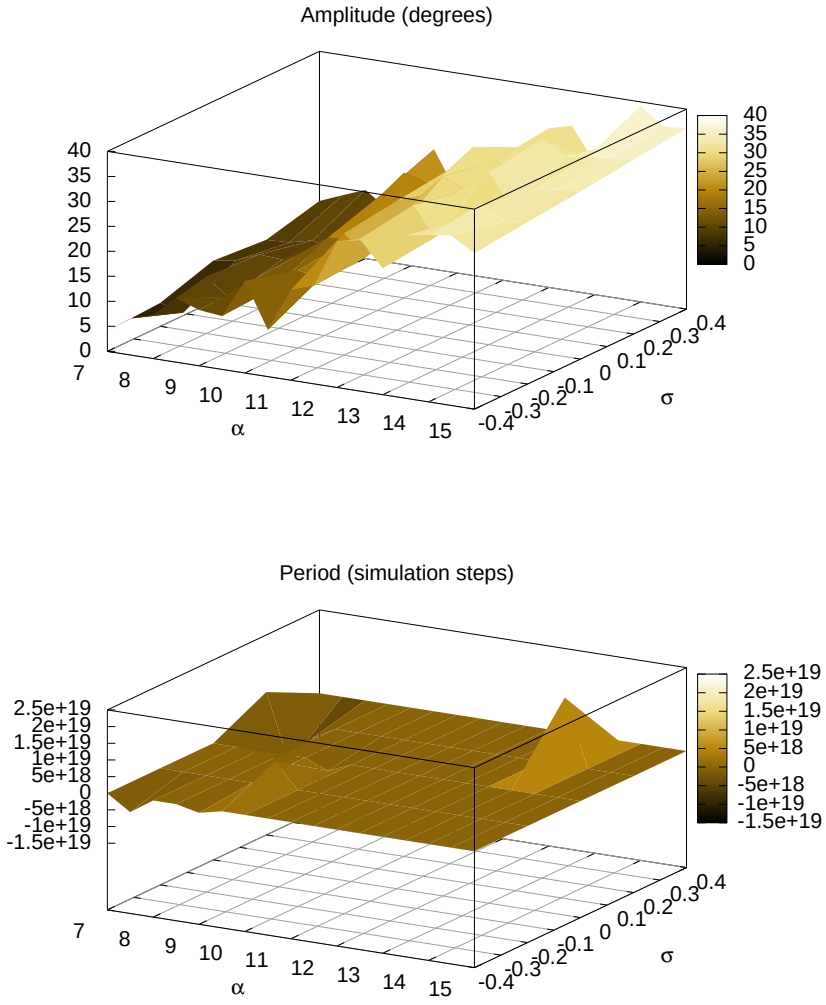


Figure 6.3: Synaptic current modeled on the slow subsystem only ($\sigma_e = 1$, $\beta_e = 0$), summary of oscillator output averaged over ten simulations for each pair of α and σ. Roughness of the surface is due to the dependence on initial conditions (see figure 6.5): some simulations converge to a stable oscillation, but some others converge to a non-oscillatory state. Since all simulations are run with random initial conditions, the actual average will depend on the proportion of well-behaved and ill-behaved simulations.

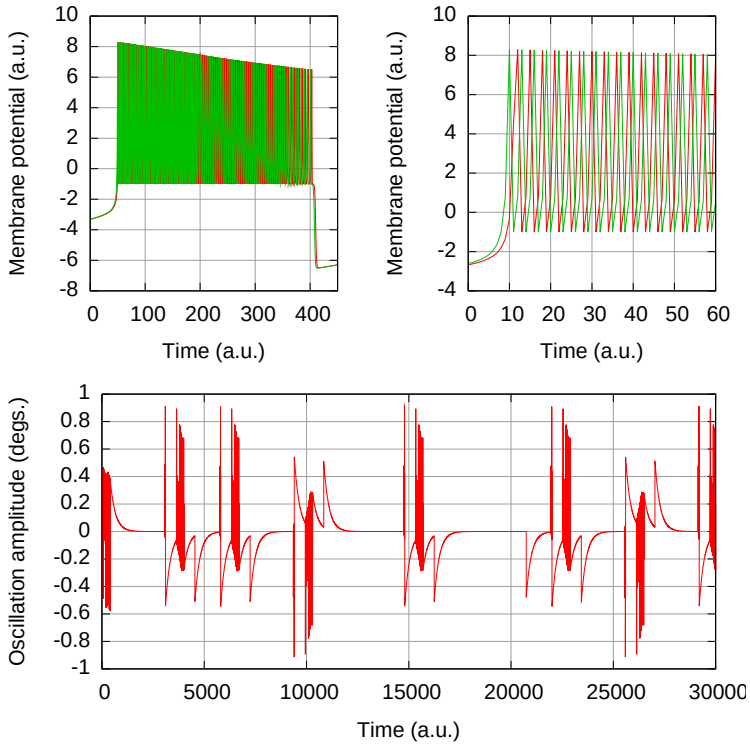
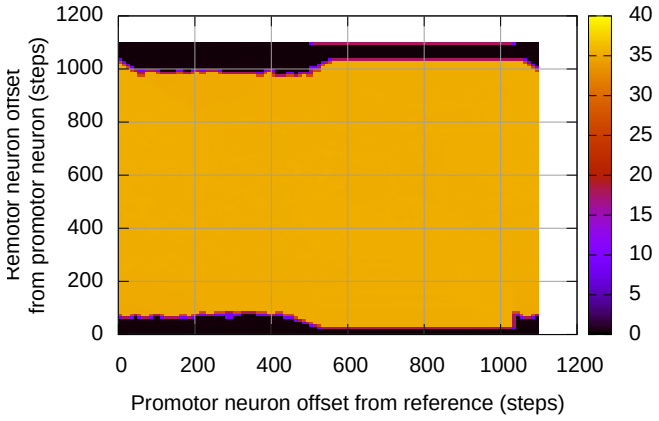
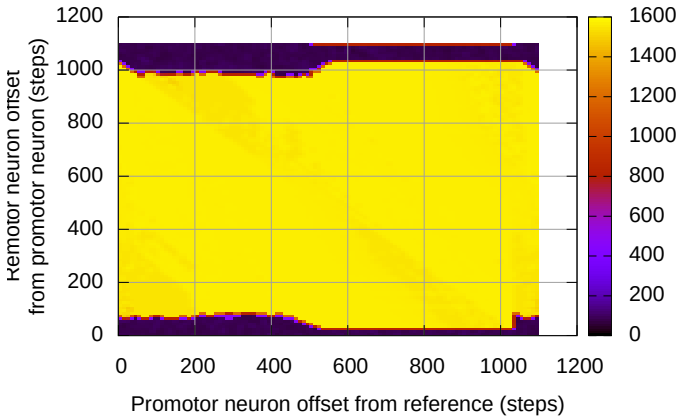


Figure 6.4: Depending on initial conditions, the half-center oscillator with slow subsystem synaptic currents can lock at a spike level instead of the burst level. The expected behavior would be anti-phase synchronization between the slow subsystems of the promotor and the remotor neurons. The observed behavior is instead in-phase synchronization between slow subsystems and anti-phase synchronization among the fast subsystems. The left panel shows the remotor and the promotor neurons bursting in-phase. The right panel shows a closeup of the same bursts showing spike level anti-phase synchronization. The lower panel shows the output of the motoneuron. There is no oscillatory behavior, and the amplitude of the signal is less than 1.



(a) Amplitude



(b) Period

Figure 6.5: Stability regions of the half-center oscillator with slow subsystem synaptic currents ($\sigma_e = 1, \beta_e = 0$). Behavior of the system is dependent on initial conditions. We choose a point on the promotor neuron's limit cycle, corresponding to the onset of a burst. The promotor neuron's state is then set to this point, and advanced in time a given offset. The remotor neuron's state is set to the same state as the promotor neuron, and then advanced a different offset. Abscissæ show promotor time offset from burst onset; ordinates show remotor time offset from promotor initial state. If remotor offset from promotor is small (depending on promotor initial state), the half-center oscillator will be locked in-phase.

the desired oscillatory behavior. This state is, however, less stable than the desired anti-phase synchronization. This means that small perturbations will push the oscillator to its expected stable orbit. Chains of oscillators connected with the patterns proposed in chapter 7 will almost always eventually converge because input from neighboring modules is usually sufficient to move an oscillator to a stable oscillation.

Synaptic currents on the fast subsystem Synaptic current in the neurons of the half-center oscillator can also be modeled in the fast subsystem only setting $\sigma_e = 0$ and $\beta_e = 1$.

The amplitude of oscillation in the top panel of figure 6.7 shows a linear dependence with α and almost no influence of the σ parameter. A linear dependence with one of the parameters will provide for better predictability and control of the amplitude of the oscillator. On the other hand, the bottom panel in figure 6.7 shows an irregular surface for the period.

The results of this configuration suggest that a regular oscillation appears when the slow subsystems of the promotor and the remotor neurons are directly coupled. More complex interactions appear when only the fast subsystems are coupled. This feature can be used to generate transient behavior to handle an unexpected situation, or to generate complex motor programs.

Synaptic currents on both subsystems Modeling synaptic currents as contributing to both the fast and the slow subsystem yields the smoothest and most predictable behavior, without dependence on initial conditions. Figure 6.8 shows a very smooth behavior, with linear dependence both on α and on σ. From a control point of view this is an advantage, because now the frequency of oscillation may be precisely controlled.

The appropriate balance between fast and slow subsystem interaction may yield more complex behavior. In these sections we have studied the cases when coupling parameters take the values 0 or 1. Using intermediate values will surely result in a combination between controllability and autonomy with a rich repertoire of behaviors.

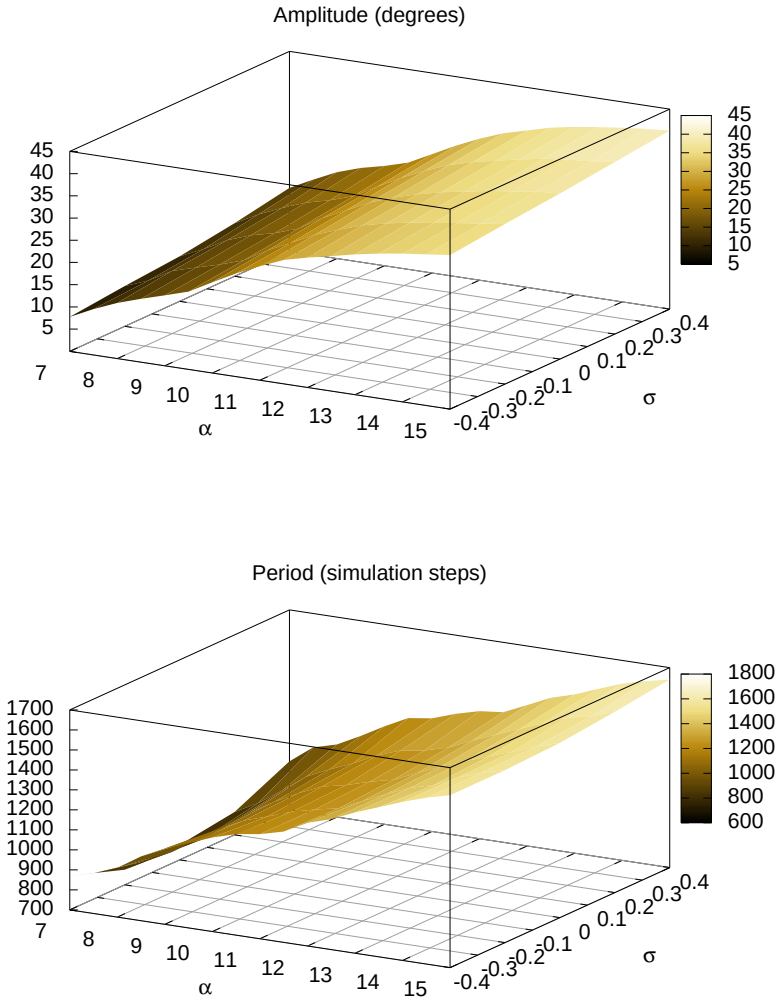


Figure 6.6: Synaptic current modeled on the slow subsystem only ($\sigma_e = 1$, $\beta_e = 0$) with controlled random initial conditions, CPG output, (top) amplitude and (bottom) period, averaged over ten simulations for each pair of α and σ. Only convergent simulations were taken into account, so the surface is much smoother and predictable than in figure 6.3. Period of the output signal is almost linearly dependent with parameter α, and slightly also linearly dependent on σ.

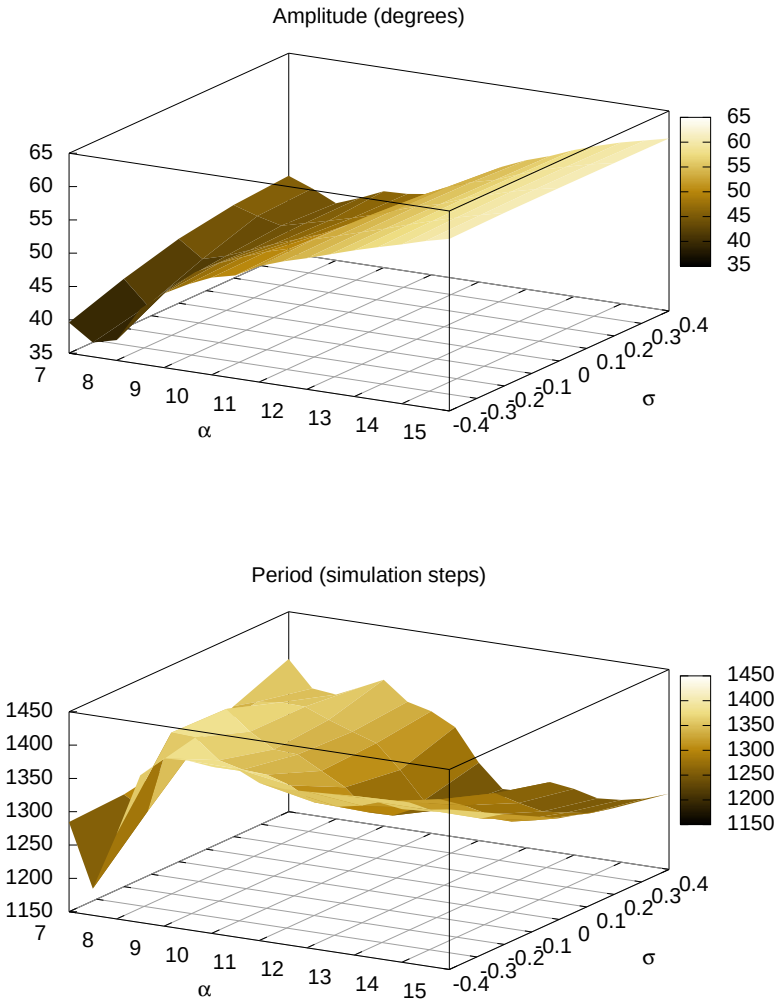


Figure 6.7: Synaptic current modeled on the fast subsystem only ($\sigma_e = 0$, $\beta_e = 1$).This configuration enables oscillatory behavior in the oscillator. However, amplitude and frequency of the generated signal are irregular. This is an evidence that suggests that stable synchronization happens when the slow subsystems of the promotor and the remotor neurons are directly coupled (i.e. $\sigma_e = 1$). Nevertheless, irregular behavior can be used to negotiate unexpected situations or to generate complex motor patterns beyond oscillation.

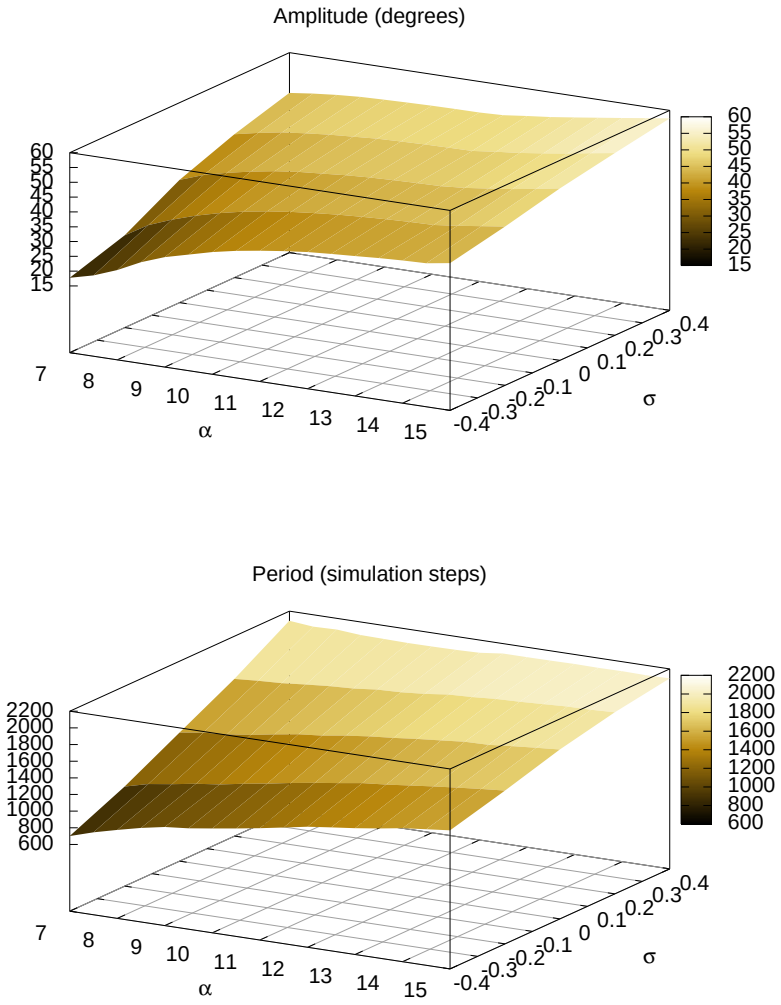


Figure 6.8: Synaptic current modeled on both the fast and the slow subsystem ($\sigma_e = 1$, $\beta_e = 1$). When the slow subsystems of neurons are directly coupled, both amplitude and period are very regular and predictable. Coupling the fast subsystems as well helps ensuring an appropriate anti-phase synchronization between neurons and avoids the bistability anomaly present in figure 6.3.

6.3.2 Autonomous adaptation to working conditions

In this section we study the ability of our oscillator to autonomously adapt to different working conditions without changing any of its parameters (Herrero-Carrón, Rodríguez and Varona, 2011). We couple a simulated servo to the oscillator and make it follow the generated pattern (figure 6.9). The oscillator will change its frequency depending on the responsiveness of the servo. We will study a wide range of responsiveness of the servo together with feedback modulation.

The structure of the analysis is similar to the previous section. We will study three different ways to incorporate feedback information into neural dynamics. For each case, a wide range of servo responsiveness and feedback modulation will be explored and amplitude and period of servo oscillation will be analyzed.

We simulate a servo using the following equation:

$$e_n = u_n - s_n \tag{6.4}$$

$$s_{n+1} = s_n + \mu e_n \tag{6.5}$$

where s_n is servo position at time step n, u_n is the control parameter indicating the target position, μ is a responsiveness constant ranging from 0 to 1, whereby values closer to 0 mean a very slow servo (up to infinitely slow, i.e., motionless if $\mu = 0$) and values closer to 1 mean a very quick servo (up to infinitely quick, with only one time step of delay if $\mu = 1$), and e_n is the position error of the servo that serves as feedback for the controller oscillator.

Coupling between the simulated servo and the CPG is performed by using $u_n = m(t)$ from (6.3) as the control parameter. Then, for the promotor and remotor neurons of the oscillator, the equation of synaptic input would read (compare to the original in (5.8)):

$$I_n^p = g \cdot r_n^p \cdot (x_n^p - E_{syn}) + Ae_n \tag{6.6}$$

$$I_n^r = g \cdot r_n^r \cdot (x_n^r - E_{syn}) - Ae_n \tag{6.7}$$

where 'p' and 'r' denote whether the postsynaptic neuron is the promotor or the remotor neuron respectively, and A is a scaling factor that represents the importance of feedback.

In the following sections we will repeat the procedure of section 6.1. We will perform the same set of experiments for the three different synaptic configurations.

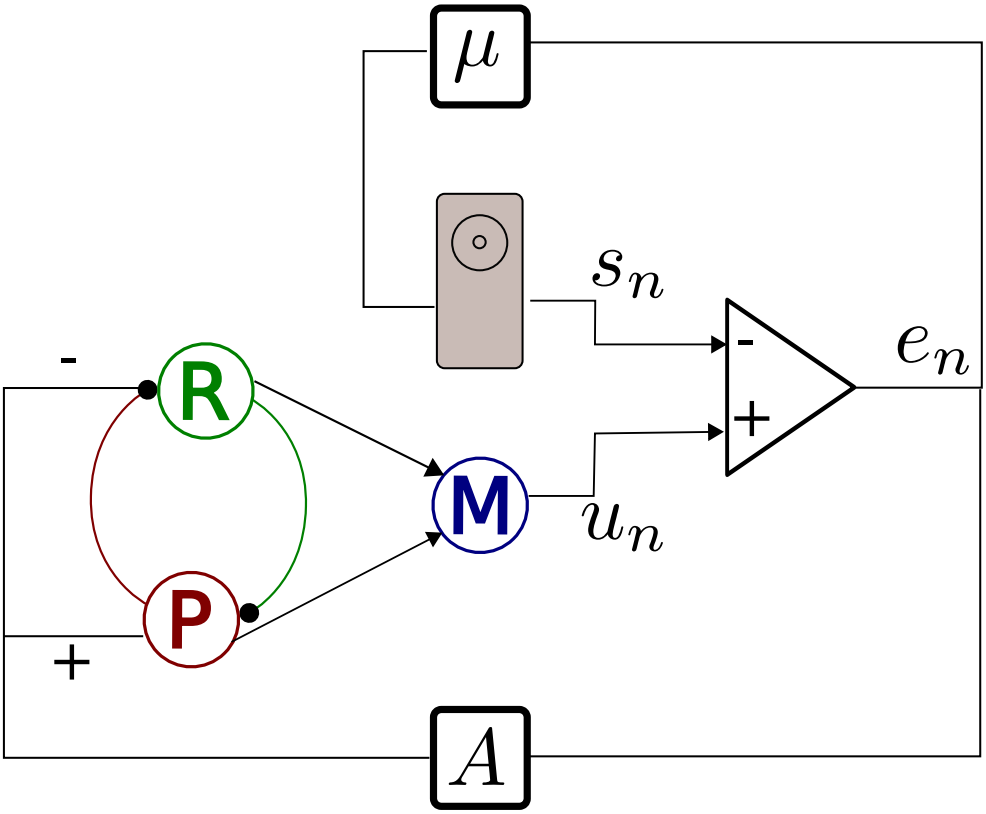


Figure 6.9: Entrainment setup diagram. Our oscillator can use error feedback from the controlled servo to adapt its frequency to servo response. We simulate a servo with position s_n and a proportional controller with a response constant μ. The error e_n is calculated as the difference between the reference set by the oscillator u_n and the actual position of the servo. A feedback gain term A multiplies the error, and the result is injected as a synaptic current to the neurons, with the same sign to the promotor neuron and opposite sign to the remotor neuron. Depending on μ, the oscillator will effectively adapt its frequency if A is sufficiently large.

Feedback on the slow subsystem Introducing feedback on the slow subsystem is an effective way to achieve adaptation. Positive feedback will generally increase the length of bursts, while negative feedback will generally decrease it. As shown in figure 6.10, this configuration can change its period over orders of magnitude from its nominal oscillation if the follower servo is too slow. Note that the lower panel in figure 6.10 ranges from 0 to 100000, an order of magnitude larger than the other two sub-figures.

The overall behavior of the oscillator, for different servo responsiveness and feedback scaling is shown in figure 6.11. Both amplitude and period remain constant in the higher range of servo responsiveness μ. In this region servos are still fast enough to follow the selected configuration of our oscillator. As responsiveness decreases, period of oscillation increases if feedback is sufficiently strong (A close to 1). As a side effect of period increase, amplitude may also increase, as seen in figure 6.11. Depending on the time constants of the motoneuron (see equation 6.3), the reference signal generated by the oscillator may saturate at different speeds. Therefore, for higher frequencies the output of the oscillator may be lower if the motor neuron dynamics are relatively slow.

However, there is a dependence on the initial conditions of the oscillator. In figure 6.5 we plot the convergence values of amplitude and period of oscillation of a very fast servo driven by our oscillator. The experiments were carried out as follows: first, a reference value for the promotor neuron was set; then, an offset in simulation steps from the reference value was selected, and the promotor neuron was simulated forward that time; the remotor neuron was set to exactly the same state of the promotor neuron at this point; finally, an offset in simulation steps from this state was chosen for the remotor neuron, and it was advanced accordingly. To sum up, we study the effect of initial conditions of the neurons, and offset between them at the beginning of the simulation. The results clearly show two stable regions: one with stable oscillations and one with no rhythmic activity.

Selected examples of the above results are shown in figure 6.4. If neurons begin sufficiently close from one another, the result is an in-phase synchronization activity. Anti-phase synchronization, however, appears at a spike level. That is, while bursts happen simultaneously, intra-burst spikes happen alternately.

Feedback on the fast subsystem While modeling feedback as a term of the fast subsystem achieves anti-phase synchronization of promotor and remotor neurons, adaptability is now compromised. Figure 6.12 shows how, even though the CPG still generates oscillations, period variation is now much more restricted than in the previous case. Note that in the lower panel

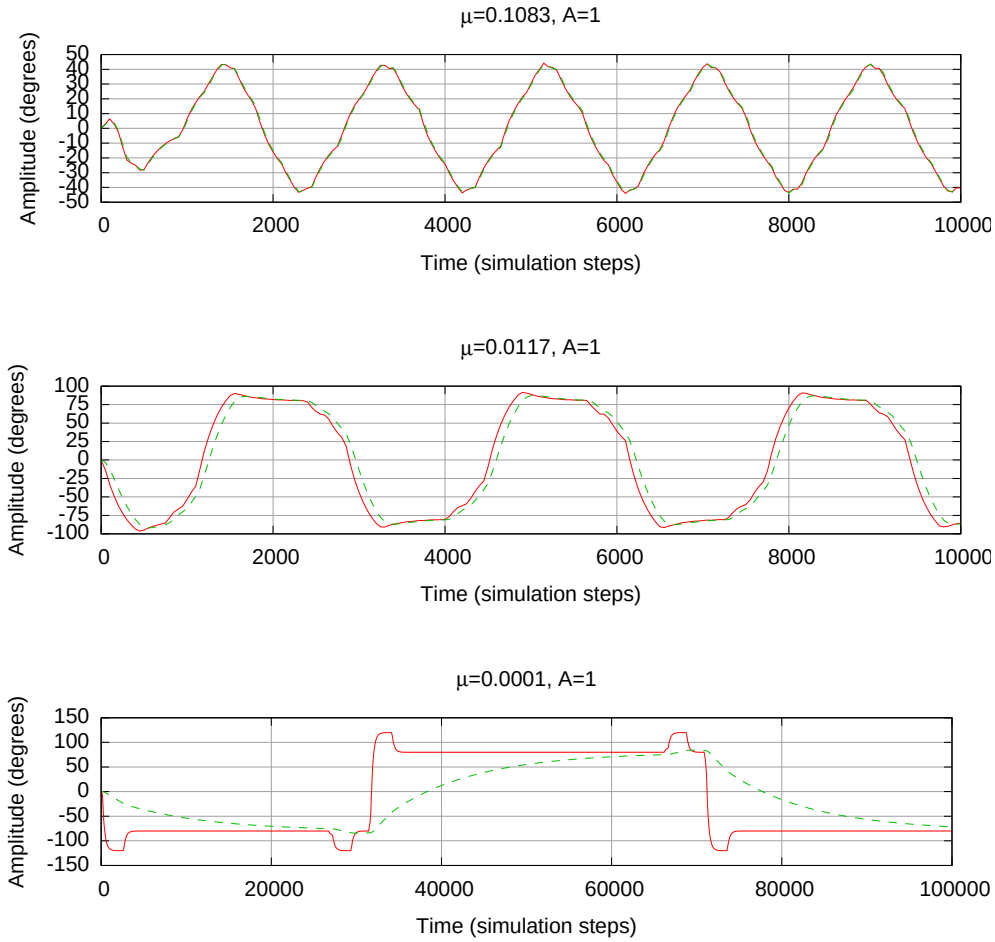


Figure 6.10: Feedback on the slow subsystem only: output the half-center oscillator and entrained servo. Output of the oscillator is plotted in red, solid lines; servo response is plotted in green, dashed lines. (Top) If the servo is fast enough to follow the oscillator, the oscillator works at its nominal regime, with an amplitude of 40 degrees and a period of approximately 1600 units of time. (Center) Our oscillator gracefully adapts, without changing any of its parameters, to a slightly slower servo. As a side effect of sustained neuron activity the amplitude raises above the amplitude in the top panel. Period is almost double as in the top panel. (Bottom) Our oscillator is able to sustain its oscillations well over orders of magnitude (notice the change of scale of abscissæ) in case the servo cannot follow it on time.

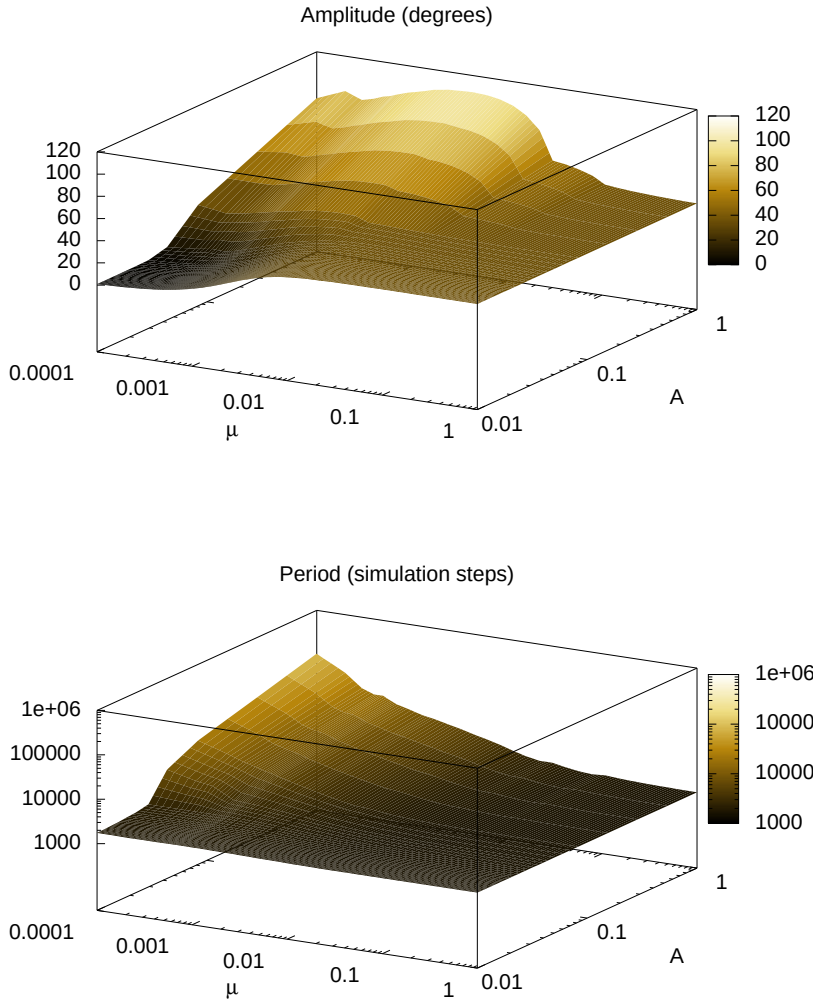


Figure 6.11: Mean amplitude (top, in degrees) and period (bottom, in time steps) of oscillation of a servo in an entrained setup, depending on servo responsiveness and feedback strength (note logarithmic scales in all axes). μ (in a.u.) indicates the responsiveness of the servo, ranging from 0 (motionless) to 1 (infinitely fast servo); A is a dimensionless scaling factor for feedback, ranging from 0, no feedback at all, to 1, the error is fed back to the oscillator. If the servo is fast enough (in the region $\mu \in [0.1, 1]$) the period of oscillation does not depend on feedback strength. For slower servos, feedback strength is relevant in how long the oscillator will *wait* for it to reach the target position: for lower values of A, the oscillator oscillates at its nominal frequency due to insufficiently strong feedback, while for higher values of A, the position error of the servo forces the oscillator to a lower frequency.

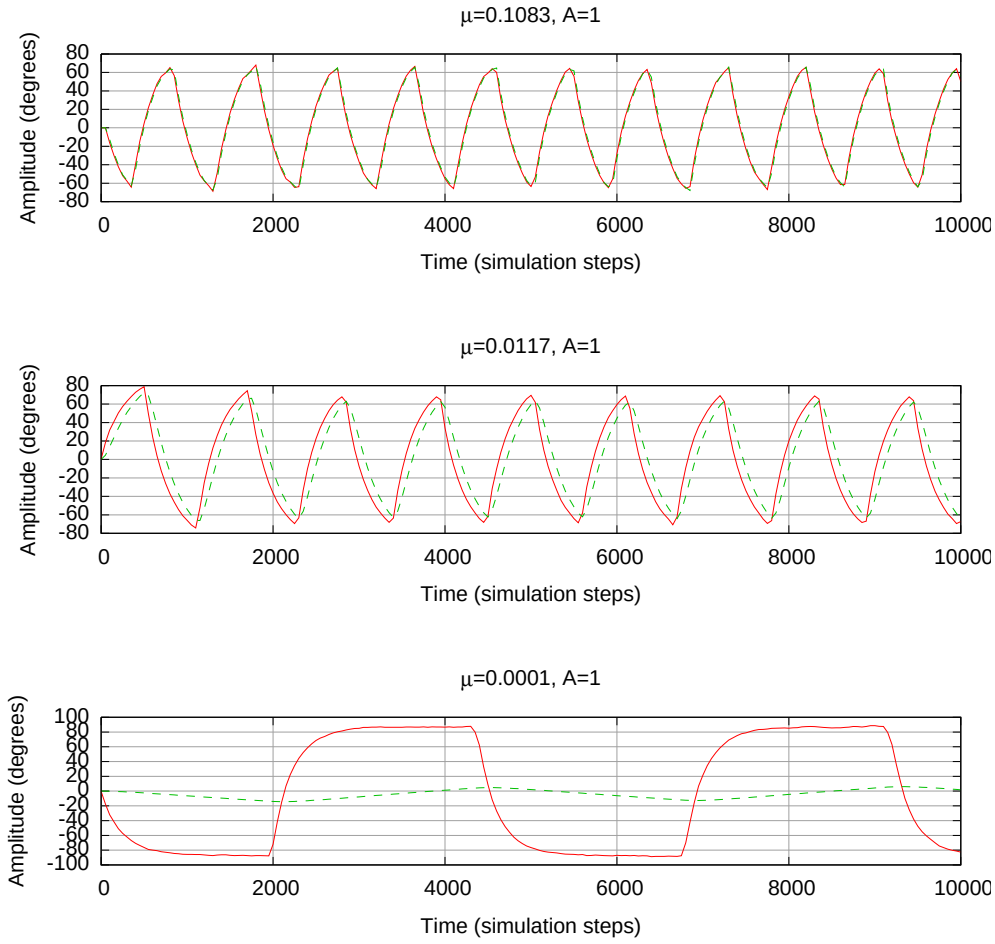


Figure 6.12: Feedback on fast subsystem only: output of the oscillator and entrained servo. Output of the oscillator is plotted in red, solid lines; servo response is plotted in green, dashed lines. (Top) If the servo is fast enough to follow the oscillator, the oscillator works at its nominal regime, with an amplitude of 60 degrees and a period of approximately 900 units of time. (Center) In this case, the oscillator slightly reduces its oscillation frequency, but not enough for the servo to completely follow it. Servo amplitude is slightly lower than oscillator amplitude. (Bottom) In the case of a very slow servo, our oscillator reduces its period considerably, but not enough for an effective control of the servo. The amplitude of the servo is considerably smaller than that of the oscillator, which means that this configuration cannot sustain oscillations for a long time (note that in this case the abscissa ranges from 0 to 10.000 instead of 100.000, as in the other two sections).

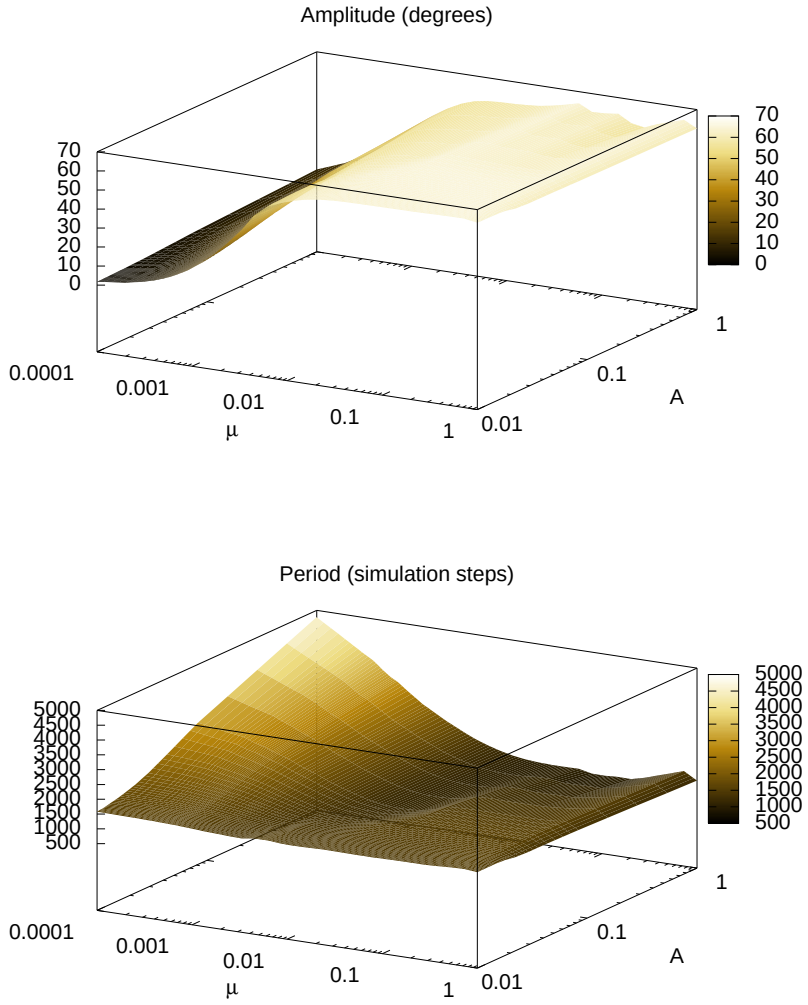


Figure 6.13: Mean amplitude (top, in degrees) and period (bottom, in time steps) of oscillation of a servo in an entrained setup, depending on servo responsiveness and feedback strength (note logarithmic scales in all axes). μ (in a.u.) indicates the responsiveness of the servo, ranging from 0 (motionless) to 1 (infinitely fast servo); A is a dimensionless scaling factor for feedback, ranging from 0, no feedback at all, to 1, the error is fed back to the oscillator. If the servo is fast enough (in the region $\mu \in [0.1, 1]$) the period of oscillation does not depend on feedback strength. For slower servos, feedback strength is relevant in how long the oscillator will *wait* for it to reach the target position: for lower values of A, the oscillator oscillates at its nominal frequency due to insufficiently strong feedback, while for higher values of A, the position error of the servo forces the oscillator to a lower frequency.

of figure 6.12, the time axis ranges from 0 to 10000, instead of 100000 as is the case in the lower panel of figure 6.10.

Feedback on both subsystems This approach has features in common with the two previous sections. First of all, the oscillator again sustains oscillation amplitude over a wide range of frequencies, given that feedback is strong enough. Second, all initial conditions drive the oscillator to a stable oscillation, unlike in figures 6.11, where dependence on initial conditions may yield non-oscillatory activity.

Qualitatively, the behavior shown in figures 6.14 and 6.8 is very similar to that shown in figure 6.10 and 6.3, respectively. In this case, period of oscillation depends both on servo responsiveness μ, as well as feedback scaling A.

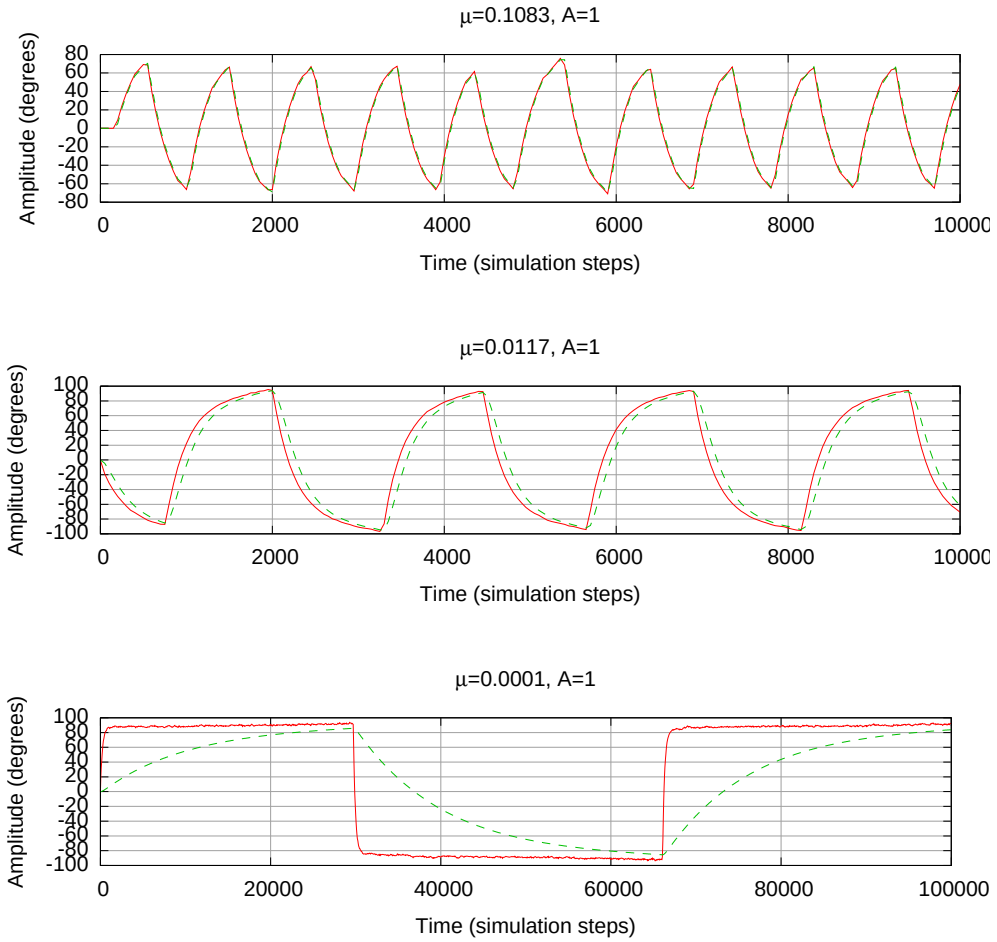


Figure 6.14: Both fast and slow subsystems: output of our oscillator and entrained servo. Output of the CPG is plotted in red, solid lines; servo response is plotted in green, dashed lines. (a) If the servo is fast enough to follow the oscillator, the CPG works at its nominal regime, with an amplitude of 40 degrees and a period of approximately 1600 units of time. (b) Our CPG gracefully adapts, without changing any of its parameters, to a slightly slower servo. As a side effect of sustained neural activity the amplitude raises above the amplitude in the previous case. Period is almost double as in the previous case. (c) Our CPG is able to sustain its oscillations well over orders of magnitude (notice the change of scale of the abscissa) in case the servo cannot follow it on time.

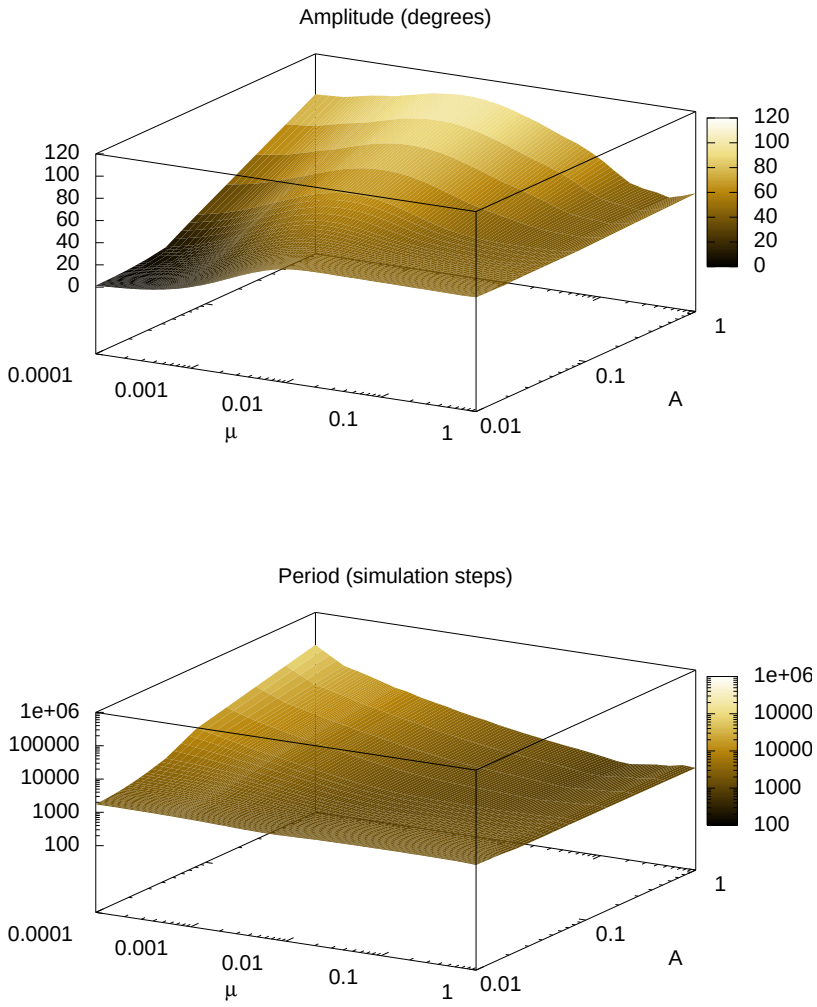


Figure 6.15: Mean amplitude (top, in degrees) and period (bottom, in time steps) of oscillation of a servo in an entrained setup, depending on servo responsiveness and feedback strength. μ indicates the responsiveness of the servo, ranging from 0 (motionless) to 1 (a.u.), an infinitely fast servo; A is a scaling factor for feedback, ranging from 0, no feedback at all, to 1 (a.u.), the error is fed back to the CPG. Clearly, if the servo is fast enough (in the region $\mu \in [0.1, 1]$) the period of oscillation does not depend on feedback strength. For slower servos, feedback strength is definitely important in how long the CPG will *wait* for it to reach the target position: for lower values of A, the CPG is oscillating at its nominal frequency due to insufficiently strong feedback, while for higher values of A, the position error of the servo is taken into account, as is the case in figure 6.11.

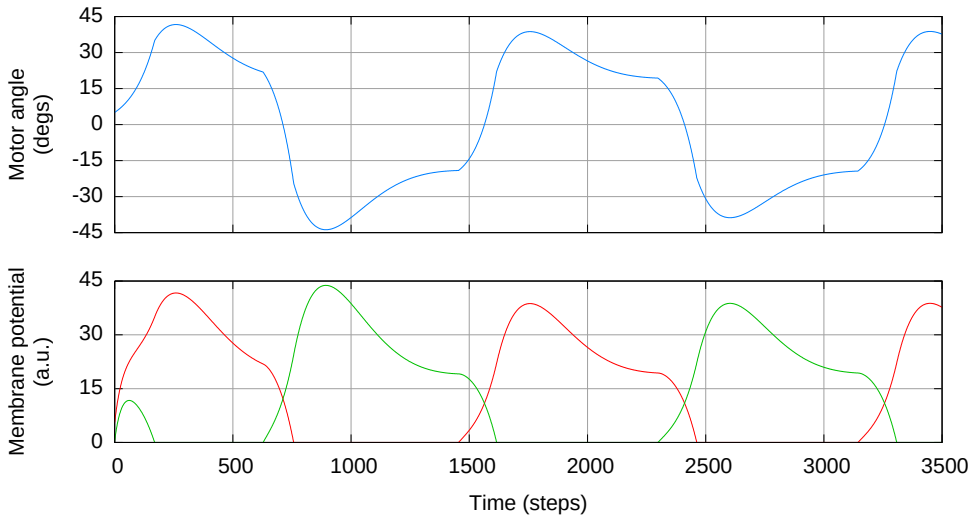


Figure 6.16: Sample activity of Matsuoka's oscillator. Upper panel: output signal to control a 1DOF joint; Lower panel) output of each one of the neurons, promotor in red, remotor in green. Parameters for the oscillator: $\beta = 2.5; \tau_1 = 0.112; \tau_2 = 0.56; C = 70; w = 2.5$.

6.3.3 Robustness against noise

An oscillator in a real working environment will be a subsystem of a bigger system. It will interact with other oscillators, with parts of the robot and many other subsystems. We have shown that our oscillator is flexible and can be configured to work in a wide range of regimes. Now, we want to explore how robust it is in the presence of noise (Herrero-Carrón et al., 2010*c*). For this, we will inject noisy currents into the neurons and measure their effect on amplitude and period of oscillation. We will also compare it to another commonly used nonlinear oscillator[1].

A popular non-linear oscillator: Matsuoka

Let us examine the exact architecture and behavior of Matsuoka's oscillator and the one we propose. As explained in section 6.1, there are two neurons that mutually inhibit each other. Each one of them is capable of generating rhythmic activity in isolation, and when coupled together they achieve anti-phase synchronization. This behavior can be observed in figure 6.16 for Matsuoka's oscillator.

[1]Video footage is available in our website `http://arantxa.ii.uam.es/~gnb/partX.avi`, where X is 1,2,3 or 4

Mathematical description of Matsuoka's oscillator

In Matsuoka's oscillator, each individual neuron is described by the following equation:

$$\dot{x} = (-x - \beta v - I + C)/\tau_1 \tag{6.8}$$
$$\dot{v} = (-v + y)/\tau_2 \tag{6.9}$$
$$y = \max(x, 0) \tag{6.10}$$
$$I = wy' \tag{6.11}$$

where x and v are the internal variables of the neuron and y is its output. Coupling between the two neurons is realized using parameter I, which represents external input, and is equal to the output of the other neuron multiplied by some weighting factor. Finally, the ensemble output of the circuit is:

$$m(t) = y_p - y_r \tag{6.12}$$

where $m(t)$ is the motor signal that will drive the joint, y_p is the output of the promotor neuron and y_r is the output of the remotor neuron. See figure 6.16 for a sample trace.

Comparing robustness against noise

If an oscillator is coupled to an external input to achieve entrainment, noise coming from environmental sources may severely degrade the performance of the controller. Thus an oscillator controlling one joint has to be flexible enough to adapt to valid inputs, but robust enough to not be disturbed by noise. We want to study how a very popular oscillator model, namely Matsuoka's and our proposed model behave when faced with noisy input.

We will assume that the joint being controlled is perfect, so error between the desired position and actual position is always zero. We will then assume that measures are noisy, with a gaussian distribution of amplitude. We will study how standard deviation of this distribution affects each oscillator.

It is difficult to set up the scenarios so that measures taken on one oscillators can be compared and evaluated against the other oscillator. All the more so taking into account that one is formulated using differential equations and the other one is a map.

The first step has been to use an Euler integration method to integrate Matsuoka's oscillator with a time step of 0.001 time units. Then, we have chosen the time constants that achieve a frequency similar to our oscillator, measured in simulation steps. Details for the parameters for Matsuoka's oscillator can be found in table 6.1. Similarly, details for our oscillator are

Table 6.1: Parameters for Matsuoka's oscillator

Parameter	**Value**
β	2.5
τ_1	0.112
τ_2	0.56
C	18 and 40
w	2.5

Table 6.2: Parameters for our oscillator

Parameter	**Value**
α	15
σ	-0.33
μ	0.001

(a) Neuron model parameters

Parameter	**Value**
λ	0.5
β	5
θ	0
E_{syn}	9
g_{syn}	10
$[T]$	1
t_r	0.01

(b) Synapse model parameters

presented in table 6.2. In our oscillator, amplitude of the oscillation can be scaled in different ways. One possibility is to tune parameters α and σ of the neurons as shown in section 6.3.1. Another possibility is to adjust the time constants of the motoneuron in equation 6.3. The first method will actually change the qualitative dynamics of the oscillator. Higher values of α and σ induce longer bursts and a different time distribution of the spikes. While the slow subsystem of neurons will still negotiate an effective rhythm, the shape of the motor signal will be different, since it is encoded in spike distribution. On the other hand, changing time constants of the motoneuron will effectively perform a scaling of the signal without affecting the dynamics of the oscillator.

Similarly, parameter C in Matsuoka oscillator (see equation 6.11) performs a scaling of amplitude but the intrinsic structure of the oscillator does not change. Nevertheless, we have studied two variants of Matsuoka's oscillator to assess the effect of this parameter.

To analyze how each oscillator responds to noise we have run ten simulations for each different value of standard deviation of noise. Each simulation takes 100 000 simulation steps, which accounts for about 70 oscillations. At

each simulation step each one of the two neurons is injected a different sample of a gaussian distribution with mean zero and standard deviation ranging from 5 to 50. Random distributions are simulated using the Mersenne twister pseudo-random number generator from the boost random library[2], seeded with a random value taken from `/dev/random` on a linux machine. To inject noise into a neuron, each random sample is added to the corresponding I term in each step (see (6.11) and (5.7)).

We propose an intuitive way to assess if noise has equal impact, in terms of amplitude, on one system and the other. In each one of the oscillators, each neuron is defined by a two dimensional system of the form:

$$x_{n+1} = F_x(x_n, y_n) \tag{6.13}$$
$$y_{n+1} = F_y(x_n, y_n, I_n) + k\eta_n \tag{6.14}$$

where I_n is external input and η_n is noise from measurement at time n. In each one of the simulations performed we have studied the relationship of the term $k\eta_n$ to $F_y(x_n, y_n, I_n)$ in (6.14). For this, we have taken the sequence η_n and the sequence $F_n = F_y(x_n, y_n, I_n)$. We have then defined the quantity SNR (system to noise ratio):

$$SNR = 10 * \log\left(\frac{P_F^2}{P_\eta^2}\right) \tag{6.15}$$

where P_X is a measure of average signal power

$$P_X = \frac{1}{l}\sqrt{\sum X_n^2} \tag{6.16}$$

being l the length of X_n.

This measure evaluates how big the noisy perturbation is in relationship to the *diameter* of the oscillation. Our hypothesis is that for the same level of noise, oscillators with larger amplitude will suffer a lower impact from the perturbation. Indeed, figure 6.17 shows that Matsuoka oscillator with different values for its C parameter yield different SNR measures. To ensure a fair comparison we have chosen the parameters for Matsuoka oscillator to closely match the frequency, amplitude and SNR of our oscillator. See in figure 6.17 that setting $C = 18$ for Matsuoka yields similar results to the chosen configuration of our oscillator.

[2] `http://www.boost.org/doc/libs/1_42_0/libs/random/index.html`

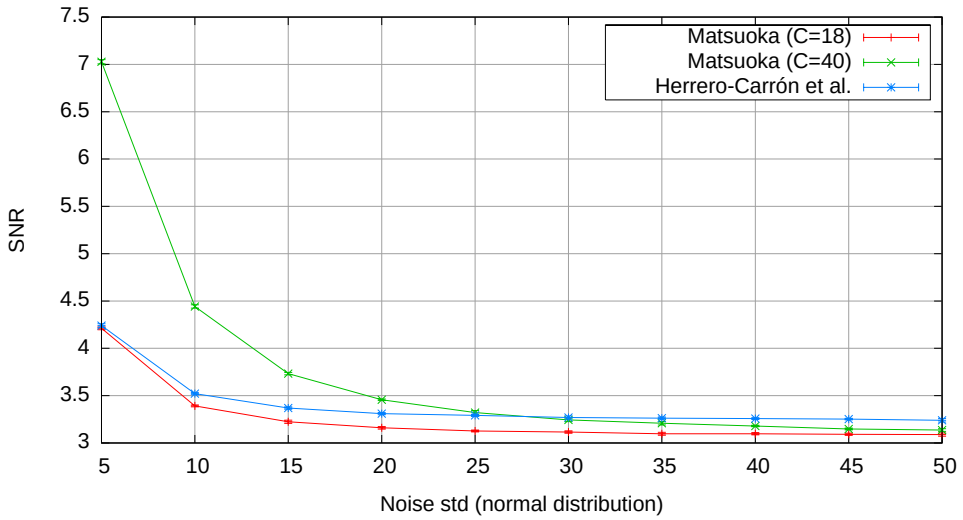


Figure 6.17: SNR (system to noise ratio) as defined in (6.15) of the two oscillators against standard deviation of a gaussian noise. Values shown are the mean of ten simulations for each value of standard deviation of the noise, error bars show standard deviation within the ten simulations. Oscillations with larger amplitude will be less perturbed for the same amount of noise. Different values of C in Matsuoka's model yield different amplitudes and different SNR values. Setting $C = 18$, Matsuoka oscillator oscillates with a similar amplitude and SNR to the configuration of choice for our oscillator.

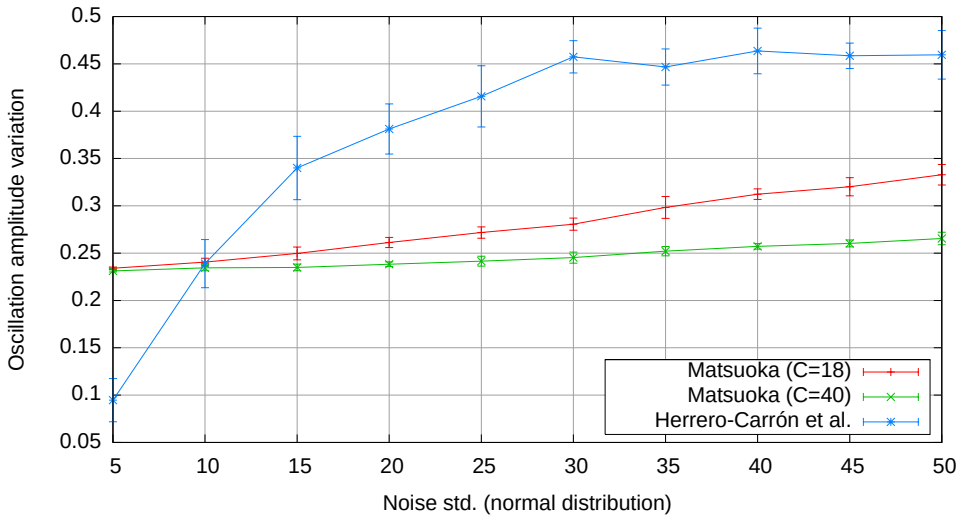


Figure 6.18: Coefficient of variation (σ/μ) of oscillation amplitude in response to increasing standard deviation of a gaussian noise. Since oscillators are not perfect sine waves, their amplitude, as calculated in (5.11) will not be constant, i.e., for low or null noise, variation will be different to zero, with a value only attributable to the shape of the signal. The two Matsuoka oscillators differ because larger amplitudes are able to accommodate larger levels of noise. Comparing Matsuoka (C=18) and our oscillator, it is clear that our oscillator performs worse in keeping a constant amplitude.

Amplitude and frequency response to noise

In order to analyze the signal generated by an oscillator, we first generate an analytic signal that uniquely represents it. The basic tool for our analysis is the Hilbert transform (see section 5.3).

In figure 6.18, both instances of Matsuoka's oscillator reveal better maintenance of phase amplitude. Looking at figure 6.20 we can see that, even though signals look very noisy for high levels of noise, the overall envelope of the oscillation is very stable. Looking at figure 6.21 we can see that the amplitude envelope suffers more degradation than in Matsuoka's. The qualitative response is different for both types of oscillators. While the two Matsuoka models differ only in a constant factor, our oscillator reaches a point of apparent maximum degradation at noise standard deviation 30.

When comparing frequency response, figure 6.19 reveals an increasing loss of performance in Matsuoka's models, again differing only in a constant factor. Our oscillator is able to maintain steady frequency variation values because it is able to generate smoother signals than Matsuoka's oscillator.

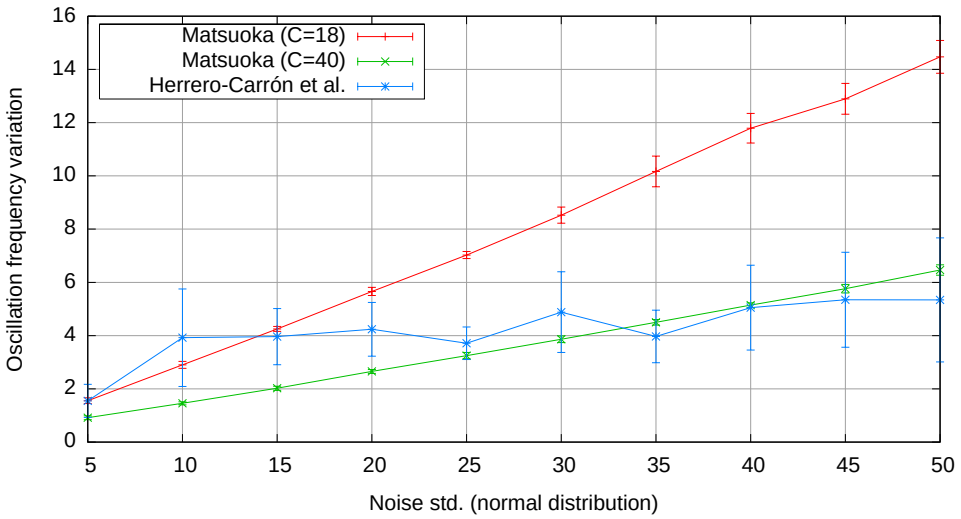


Figure 6.19: Coefficient of variation (σ/μ) of oscillation frequency from the mean in response to increasing standard deviation of a gaussian noise. The two Matsuoka oscillators show similar qualitative response, differing only in a scaling factor due to the difference in amplitudes of their oscillations. When compared to our proposed oscillator, they both show linear dependence on the amplitude of noise, while ours keeps a relative constant level of frequency. Looking at figure 6.20 and 6.21, it is evident that our oscillator always generates a smooth signal due to the filtering properties of (6.3).

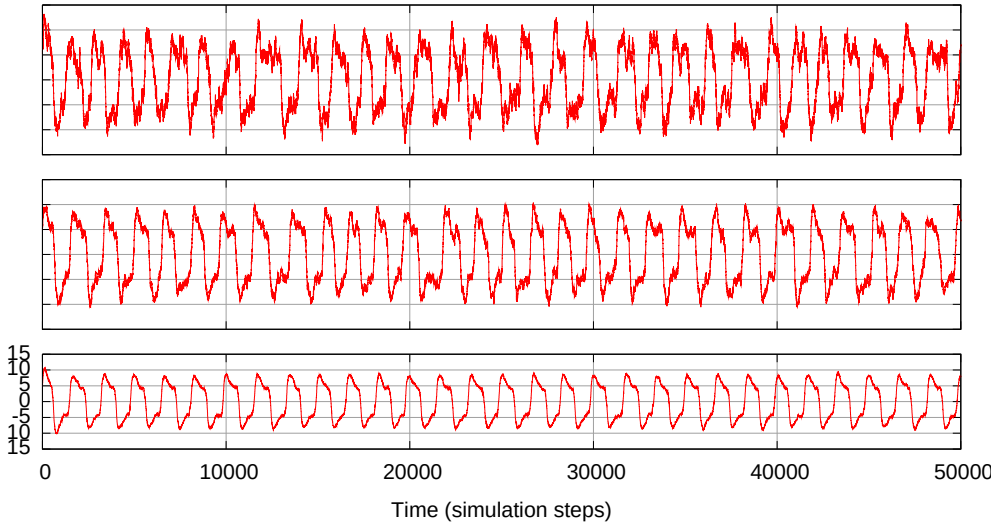


Figure 6.20: Output of Matsuoka's oscillator (C=18) for a gaussian noise with zero mean and standard deviation 5 (lower panel), 15 (center panel) and 30 (upper panel).

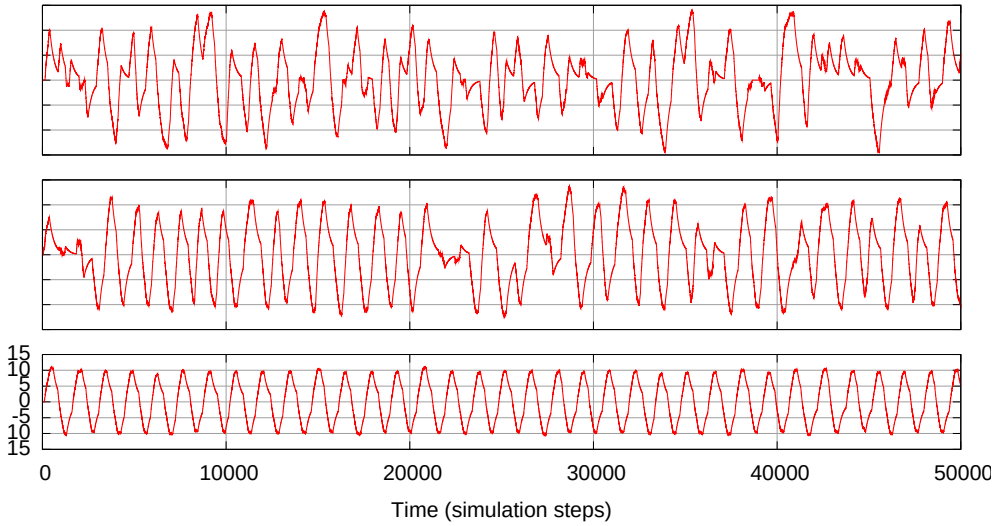


Figure 6.21: Output of our oscillator for a gaussian noise with zero mean and standard deviation 5 (lower panel), 15 (center panel) and 30 (upper panel).

6.4 Summary

This chapter introduces a modular joint controller based on a half-center oscillator built with bursting neurons and dynamical synapses. Bursting neurons produce a precise rhythm in isolation, that is very flexible when they are coupled with each other. Through mutual inhibition the two neurons of the oscillator, promotor and remotor, negotiate a stable overall pattern resulting in stable oscillations. This combination of intrinsic dynamics and non-open topology is the minimal implementation of winner-less competition dynamics.

From a dynamical point of view, the bursting mechanism used for the oscillator is a composition of two coupled dynamics. A slow system oscillates periodically with a given period while a fast system alternatively transitions from fast oscillations to a straight trajectory in phase space. These bifurcations are governed by the state of the slow subsystem, resulting in the typical pattern of a wave with spikes sitting on its crests.

Each subsystem plays a different role in the oscillator. While the slow subsystem is the main mechanism of synchronization and rhythm negotiation, the fast subsystem encodes locomotion information. That is, frequency of oscillation will be controlled by the slow subsystem and angular velocity and amplitude will be controlled by the fast subsystem. A third neuron, the motoneuron, decodes locomotion information and translates it into a signal that the actuator will understand.

Different parameters control the rhythm generated by the oscillator. Parameters controlling neural properties like bursting period and burst length also control the oscillator's period and amplitude. Then, different integration between the neuron and synapse dynamics yield different negotiation capabilities to the module. The result is a modular controller that can be flexibly configured to work in a wide range of frequencies and amplitudes to produce robust rhythms in every case.

The precision of the rhythms generated by the oscillator does not prevent adaptation to a changing environment. Such is the case when the controller is coupled to different joints with different responsiveness. Measuring their position error and without any change in its parameters, the oscillator will autonomously choose the working regime that will best sustain a precise oscillation considering the response of the joint. Periods of oscillation range over several orders of magnitude depending on joint responsiveness.

Finally, bursting dynamics have proved very robust against noise. In a working robot, modules are not isolated entities. Rather, they receive information from other modules and from sensory systems. It is necessary to be flexible for negotiation with valid inputs, and be robust to discard

irrelevant noisy perturbations. Our oscillator produces smooth oscillations when feedback from an ideal servo is a noisy gaussian signal. We believe that bursting neurons together with threshold dynamical synapses implement a robust event-based communication mechanism. This proves more resilient to noise since communication does not depend on the precise measurement of a given biophysical magnitude.

7 Connectivity Patterns

Using a neuron model with rich intrinsic dynamics and a simple topology based on mutual inhibition, we have built an autonomous modular oscillator. In this chapter we propose different strategies to build a CPG for a modular worm robot as a chain of modules (Herrero-Carrón et al., 2010*b*). Figure 7.1 shows the approach to a global controller.

The question addressed in this chapter is what mechanisms are effective negotiation channels for modules to share information and achieve a global agreement on an effective locomotion pattern. Given that we want to build modular chain controllers we apply the restriction of first neighbors connectivity and repeatability. These two restrictions provide for scalable patterns, and provide for flexible CPGs that fit very well with the physical scalability of the robot.

The strategies proposed are very general in nature, and work over a wide range of parameter values. We have chosen representative examples to illustrate the properties of each connectivity pattern, but they are not restricted to these values.

7.1 Uncoupled modules

In order to see what dynamical features the different proposed CPG models will have, it is necessary to have an idea about how the system works when modules are uncoupled.

The expected behavior is that all modules will oscillate at the same frequency with a constant phase difference, whose value depends only on the initial conditions. Figure 7.2 shows that this is only partly true. Even though phase difference is not constant, its drift is (save for the initial transient period). This shows that the actual frequency of oscillation of one module depends on the phase difference between the promotor and the remotor neuron within the module and the particular initial conditions of each one.

Although surprising, we can take advantage of this inter-module phase drift to study the proposed central pattern generating circuits: if inter-module phase difference is different from constantly drifting, the proposed

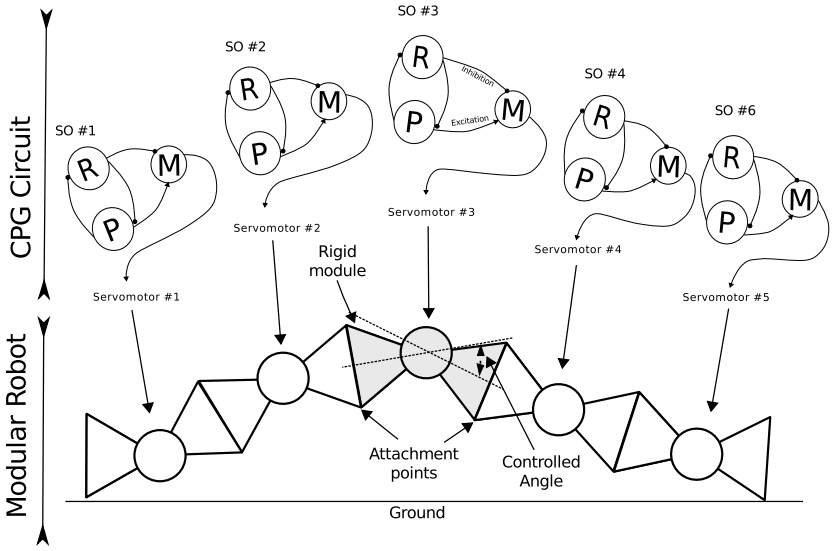


Figure 7.1: Worm modular robot and modular oscillators.

connectivity actually has an effect on phase difference.

Parameters used in the example are $\alpha = 15$, $\mu = 0.001$, $\sigma = -0.33$, $\beta_e = 0$ and $\sigma_e = 1$ for all neurons and $\lambda = 0.5$, $\beta = 10$, $\tau = 0$, $E_{syn} = 9$, $g_{syn} = 1.5$, $T = 1$ and $release\_time = 0.01$ for contralateral synapses.

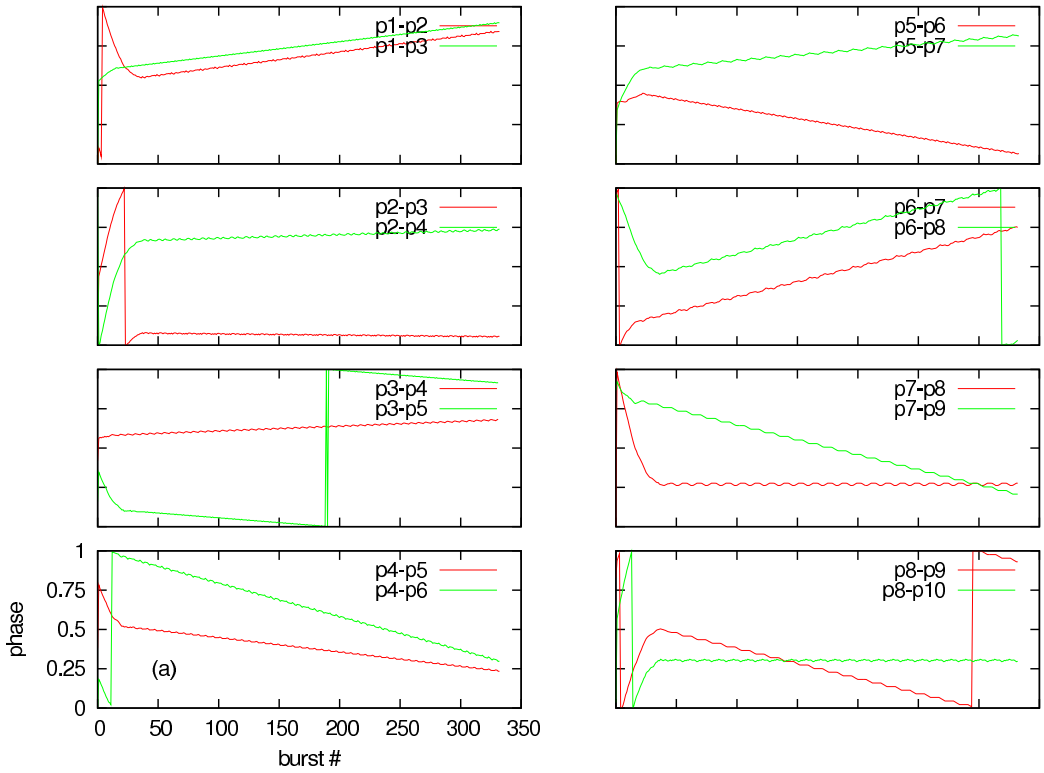


Figure 7.2: Uncoupled modules: phase differences for independently oscillating modules. There are two differentiated regimes, see for example the p8-p9 curve. There is a steep rising of the curve and then, at approximately burst #50 the curve steadily decays in a much smoother way. The first regime is due not only to frequency difference, but to a change in frequency. Both modules 8 and 9 are going through an adjustment between their promotor and remotor neurons. After this adjustment frequency settles down differently for each module, and is the only source of (constant) phase drift in the second regime.

7.2 Symmetrical inhibition, weak coupling

The first attempt was to connect adjacent modules with the same scheme as the alternating neurons within a module, that is, with two inhibitory synapses, one in each direction (two for promotors and two for remotors). These synapses have a low conductance so that their activity would influence the postsynaptic neuron's frequency but would not inhibit them completely. The idea is that inhibitory synapses should prevent adjacent neurons from firing synchronously, thus giving rise to a certain firing sequence. The question is whether this sequence would be sufficient to make the robot move in an effective manner. The pattern is shown in figure 7.3.

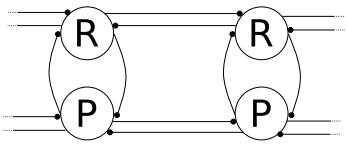


Figure 7.3: Two adjacent modules connected with the *symmetrical inhibition* model.

The parameters used in the example are: $\alpha = 15$, $\mu = 0.001$, $\sigma = -0.33$, $\beta = 0$ and $\sigma_e = 1$ for neurons and $\lambda = 0.5$, $\beta = 10$, $\tau = 0$, $E_{syn} = 9$, $g_{syn} = 1.5$, $T = 1$, $release\_time = 0.01$ for contralateral synapses.

In this particular example, the model has achieved a stable state with constant phase difference towards the end of the simulation, but this does not always happen. It remains for further investigation under which initial conditions the system arrives at a stable rhythm.

This CPG works by making small frequency adjustments. Synapses going from one module to another are weak, meaning that the effect that they have on the receptor is only small (the receptor's activity is not completely inhibited, but rather only accelerated or slowed down). As figure 7.4 shows changes in phase difference are very smooth, but very slow too.

By letting some modules oscillate faster than their neighbors, it is possible to switch the sequence in which they activate. We call this phenomenon *sequence inversion*. It is reflected as a crossing of the curves in one subplot of figure 7.4 (label (a) in the figure). Sequence inversion means that, if three consecutive modules display a propagating wave in one direction, it will suddenly start propagating in the opposite direction. The figure shows that this sequence inversion happens locally (inversion happening in three consecutive modules does not imply its happening in other three different modules).

It has not been studied under which initial conditions sequence inversion happens. Furthermore, if this phenomenon can take place in any three consecutive modules, we have no a priori control of the direction of movement of the robot (apart from setting specific initial conditions).

Advantage could be taken of this sequence inverting phenomenon if we could determine what makes it happen. Then, we could invert the direction of displacement without changing the underlying circuit.

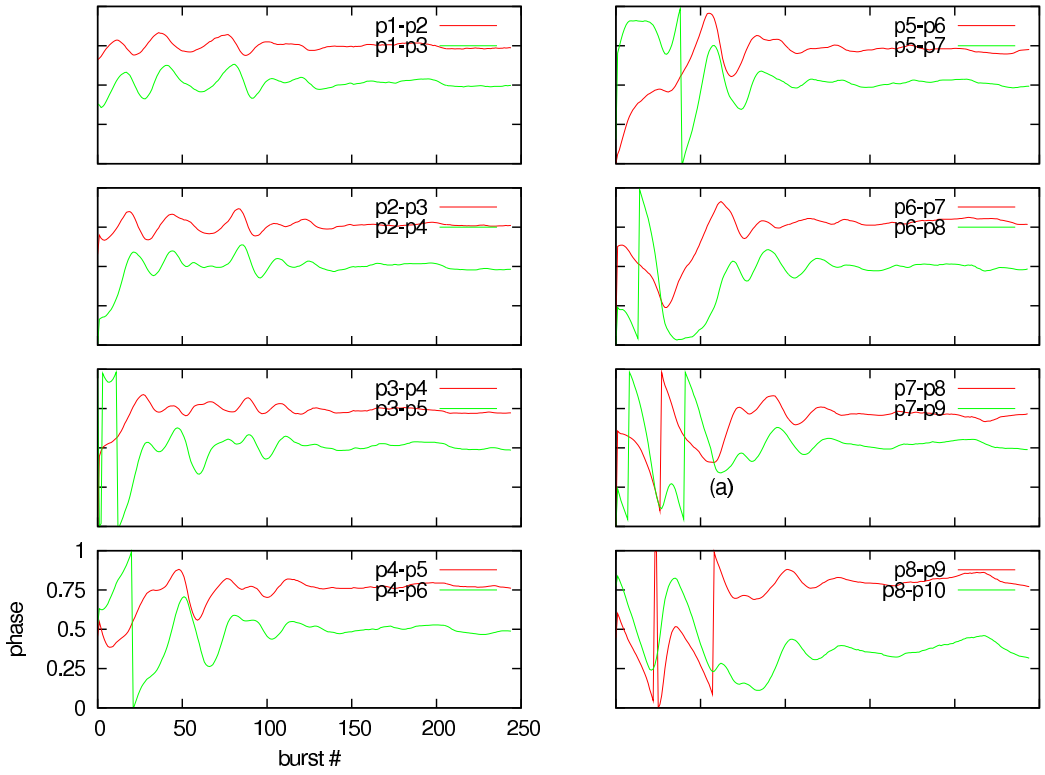


Figure 7.4: Phase differences starting with random initial conditions for the *symmetrical inhibition* model, weak coupling ($g = 0.2$). This CPG works by making small frequency adjustments. Phase difference changes smoothly until a stable state has been found. (a) This crossing of the phase curves represents a sequence inversion: module 9 is allowed to overtake module 8 in order to correct the direction of propagation of the oscillation.

7.3 Symmetrical inhibition with strong coupling

For this CPG we take the previous model and strengthen ipsilateral conductance to $g = 1.5$ (the same as contralateral), so that the receptor's activity is completely inhibited. Behavior will now depend on the recovery time of a neuron between successive inhibitions. We describe three kinds of activity: irregular, parity synchronization mode 1 and parity synchronization mode 2.

7.3.1 Irregular activity

If neuron parameters are kept as in the previous CPG but synaptic conductances are strengthened, one neuron may *dominate* its neighbors for a number of periods and then suddenly lose control. Figure 7.5 shows a sample of this irregular activity.

In this case, one neuron can be completely inhibited if its two neighbors burst simultaneously, as is the case with neuron p5 in figure 7.5. Although it cannot completely recover between successive inhibitions, it can reach an ever higher potential, finally leading it to release. This neuron can now inhibit its neighbors.

7.3.2 Parity synchronization mode 1

This mode arises when neurons do not have a chance to recover from inhibition and they cannot achieve a higher potential each time. This way, once a neuron is inhibited, it stays that way forever.

If we take three consecutive neurons: p3, p4 and p5, for instance, it might happen that p3 inhibits p4, thus releasing p5 from inhibition. Now both p3 and p5 are free from p4's inhibition and if they fire, they will inhibit, in turn, p2 and p6. If they manage to synchronize in anti-phase, it might happen that p4 never gets a chance to recover due to the total effect of alternate inhibition. This phenomenon can spread across the whole length of the system, resulting in a synchronization of all the even numbered neurons.

This kind of *parity synchronization* is found when using the following parameters: $\alpha = 20$, $\mu = 0.001$, $\sigma = -1.33$, $\beta = 0$ and $\sigma_e = 1$ for neurons; $\lambda = 0.5$, $\beta = 10$, $\tau = 0$, $E_{syn} = 13$, $g_{syn} = 1.5$, $T = 1$ and $release\_time = 0.01$ for synapses (contralateral and ipsilateral).

Figure 7.6 shows an activity sample of this kind of synchronization. As figure 7.7 reflects, almost all even neurons maintain a constant phase difference. However, figure 7.8 reveals a strange behavior between

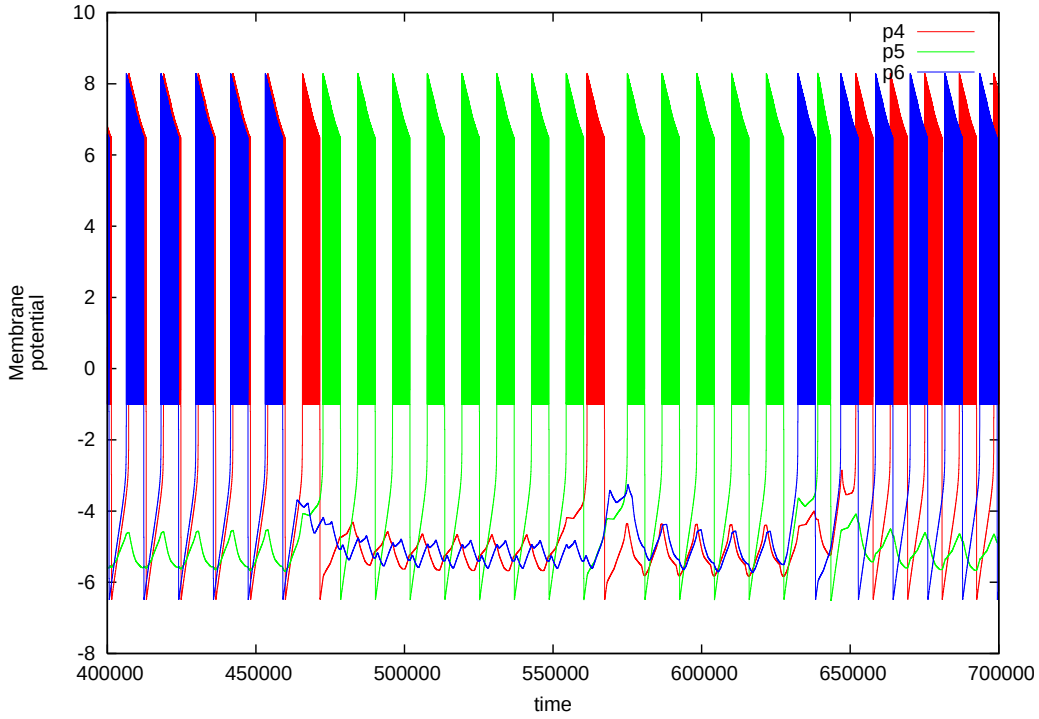


Figure 7.5: Irregular activity under symmetric inhibition, strong coupling ($g = 1.5$). Neurons can completely inhibit their neighbors. In this case there is not enough time to completely recover between successive inhibitions, but recovery speed is enough for a neuron to reach ever higher potentials. This will eventually lead the neuron to burst, thus possibly inhibiting its neighbors.

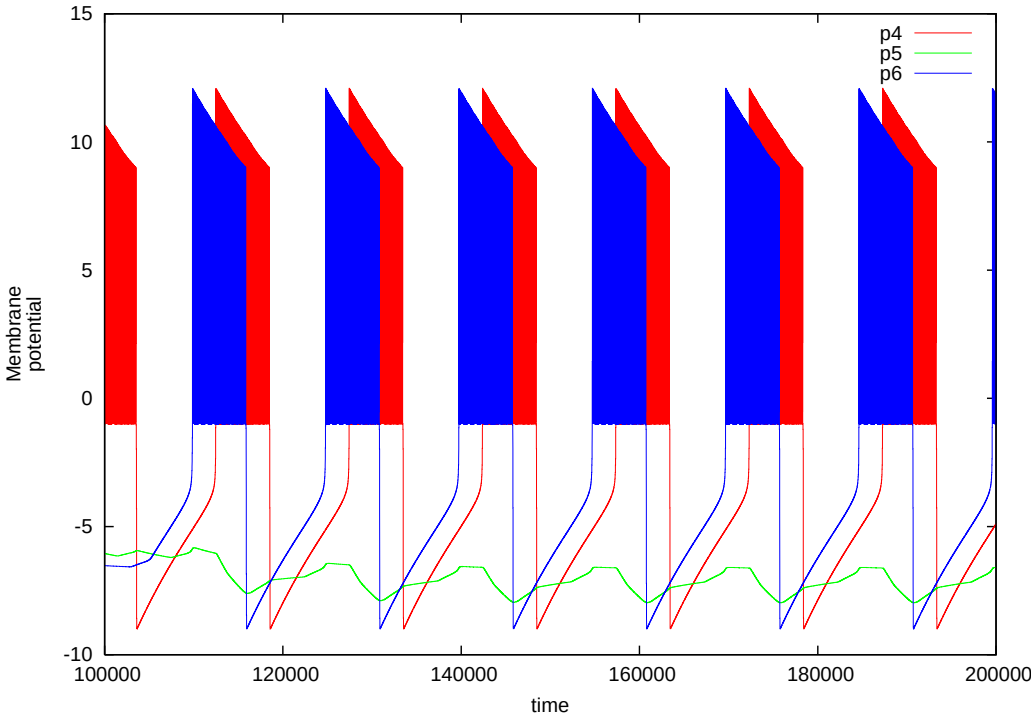


Figure 7.6: A sample of parity synchronization mode 1: odd numbered neurons oscillate approximately in anti-phase completely inhibiting even numbered neurons.

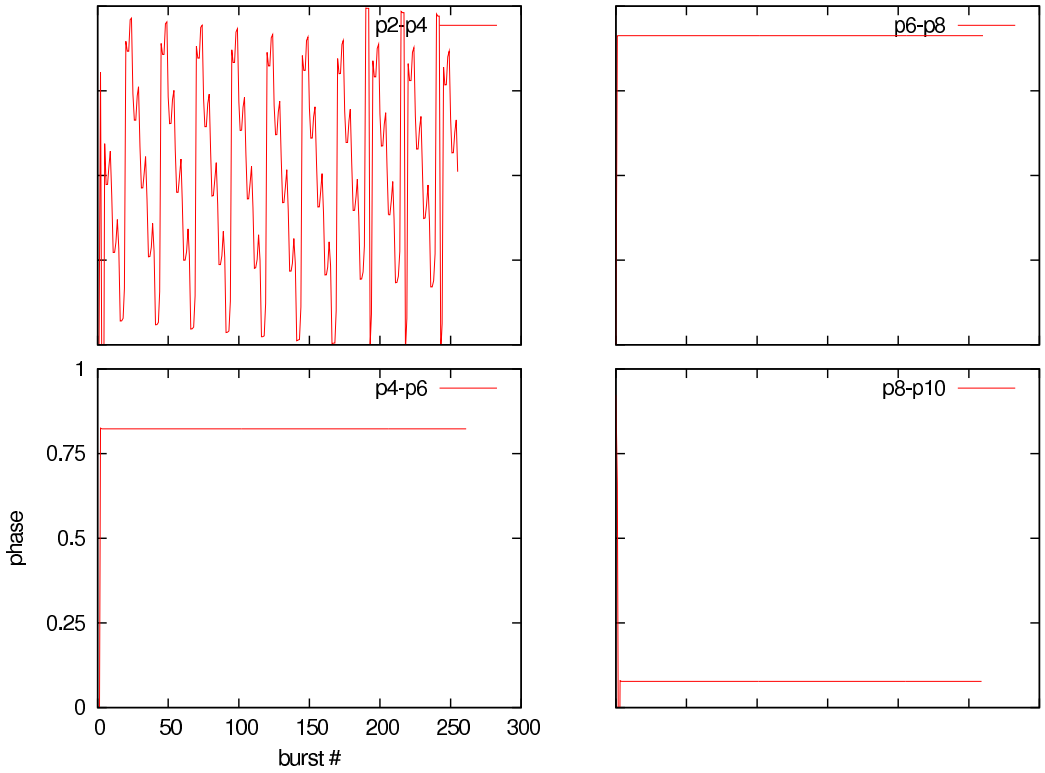


Figure 7.7: Phase differences of bursting neurons in parity mode 1: odd numbered neurons are completely inhibited by even numbered neurons (neurons not shown remain under threshold and do not burst). The activation of these neurons in different phases inhibits all other neurons completely. In this particular case, p2 does not completely inhibit p1 and it gets a chance to fire every three bursts of p2. This 3:6 frequency locking causes the irregularities seen in p2-p4.

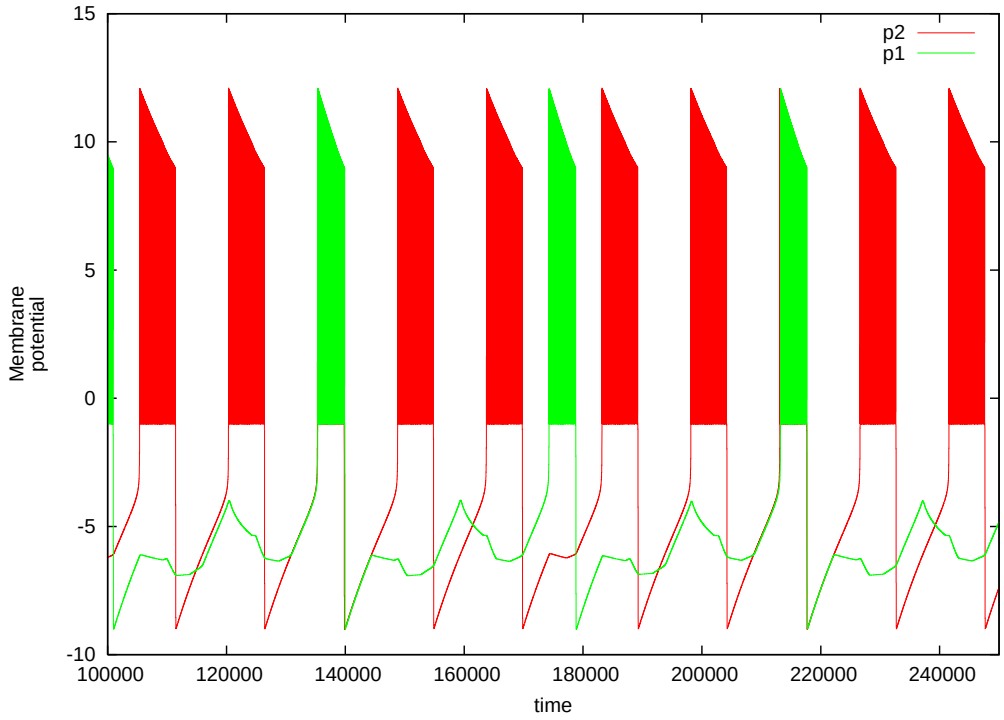


Figure 7.8: Parity mode 1: 3:6 frequency locking between neurons p1 and p2, i.e., p1 bursts three times for every six bursts of p2. This is a result of p1 being a border neuron and receiving inhibition only from p2.

7.3.3 Parity synchronization mode 2

If the inter-burst time is increased so that inhibited neurons have time to fully recover, another kind of *parity synchronization* emerges. Now, the middle neuron, p4 for instance, has time to recover from both inhibitions from p3 and p5 and, in turn, inhibits them simultaneously. After this, p3 and p5 also recover simultaneously, inhibiting p4. This time second neighbors are synchronized in phase, and in anti-phase with first neighbors.

This different kind of *parity synchronization* is found when using the following parameters: $\alpha = 6$, $\mu = 0.001$, $\sigma = -0.33$, $\beta = 0$ and $\sigma_e = 1$ for neurons; $\lambda = 0.5$,$\beta = 10$, $\tau = 0$, $E_{syn} = 3$, $g_{syn} = 1.5$, $T = 1$ and $release\_time = 0.01$ for synapses (contralateral and ipsilateral).

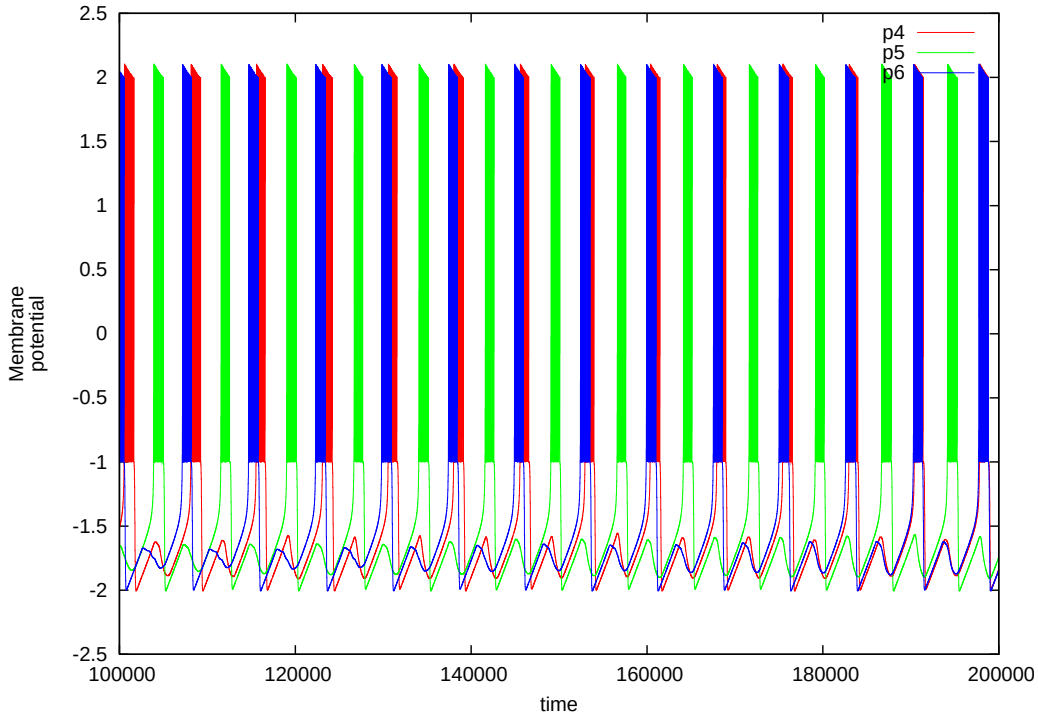


Figure 7.9: A sample of parity synchronization mode 2: neurons have time to fully recover from inhibition. In-phase synchronization with second neighbors, anti-phase synchronization with first neighbors.

7.3.4 Discussion

Symmetrical inhibition with strong coupling is not a good candidate for our purpose. The irregular activity mode is clearly not fit for a stable controller. The first mode of parity synchronization completely disables control

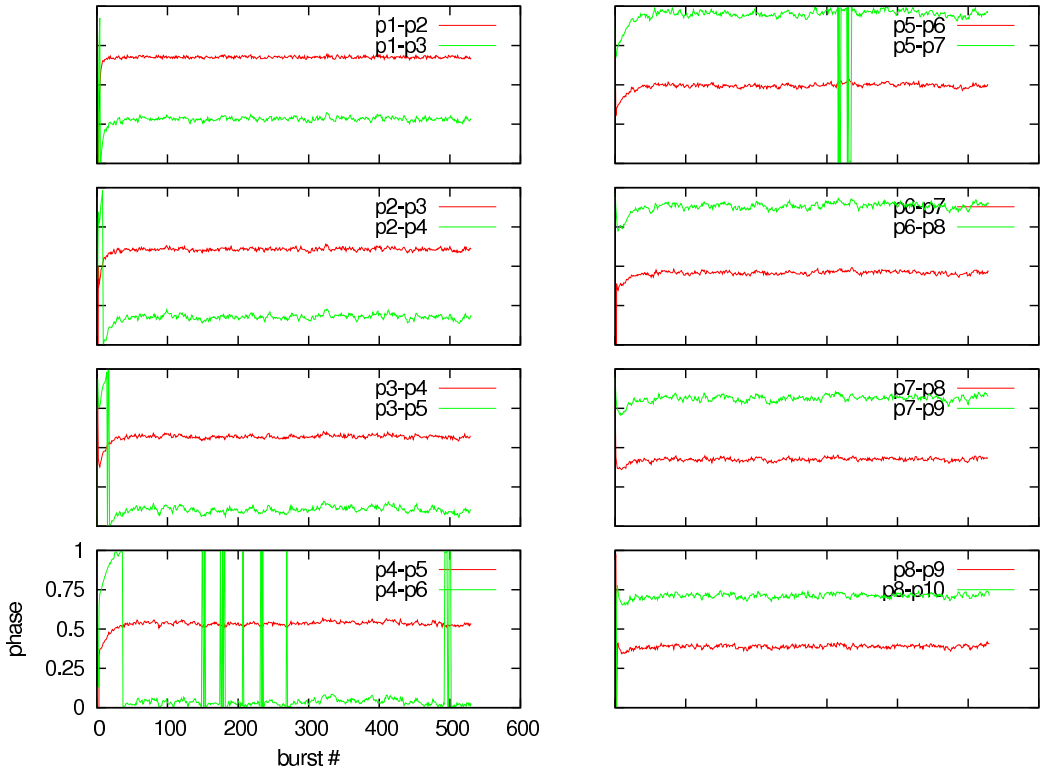


Figure 7.10: Phase differences of bursting neurons in parity synchronization mode 2: in-phase synchronization with second neighbors, anti-phase synchronization with first neighbors. (note how, although stable, phase difference is not constant across modules). Phase difference to first neighbors is approximately 1/2. This prevents a traveling wave in the body of the robot from emerging.

for half of the modules, and the second parity synchronization mode produces a standing wave rather than a propagating wave across the body of the robot.

We can extract one conclusion from the last mode of operation: phase difference between adjacent modules must be different from 0.5, otherwise a standing wave will appear rather than a propagating wave.

7.4 Asymmetrical inhibition

Using a symmetrical inhibition topology prevents predictability of the direction of locomotion and introduces a dependence on the initial conditions of the controller. However, establishing two different pathways with two different synaptic weights is an effective strategy for effective locomotion.

We have studied how different values of coupling strength affect the synchronization properties of the CPG. For each pair of coupling strengths, for the ascending and descending pathways, we have run a simulation for 4 000 000 time steps. To give an idea, the approximate wavelength of each motor signal is 1400 time steps. At the end of each simulation, phase differences between each pair of neighboring segments are calculated. To sum up each simulation we have extracted the mean phase difference of the run and standard deviation. This way we have an idea of the direction of travel, the speed of travel and the stability of the rhythm.

To illustrate how modules behave along time, we show in figure 7.11 the time evolution of their phase differences. This figure corresponds to an ascending coupling strength of 2.0 and a descending coupling strength of 4.0. As clearly seen, there is a "negotiation" period at the beginning of the simulation in which modules slowly adjust their frequency in order to find a stable phase difference. Once a stable regime is found, stable phase differences are maintained until the end of the simulation.

Figure 7.12 shows the mean phase differences for each pair of neighbor modules in only one run. Clearly asymmetric coupling forces a specific direction of travel. This is reflected in that dominant ascending coupling promotes positive phase lags and dominant descending coupling promotes negative phase lags. Symmetric coupling is not stable and hardly ever converges to a stable rhythm. Figure 7.13 shows how phase differences vary within one single run. Higher deviations from the mean value mean a lower degree of convergence or no convergence at all. Mean and standard deviation of phase differences averaged over all modules are shown in figure 7.14. This confirms the results from the previous chapter. Asymmetric weights for the ascending and the descending inhibitory pathways provide clear phase differences that result in a propagating wave of activations. Symmetric weights, on the other hand, can result in different phase-locked modes, depending on synaptic strength, or in arbitrary synchronization states depending on initial conditions.

It is interesting to see that higher values of synaptic coupling tend to push phase differences to the extremes, namely towards π and $-\pi$. A phase difference of π (which is actually the same as $-\pi$) between modules of the robot would prevent it from traveling. Rather, the body of the worm would

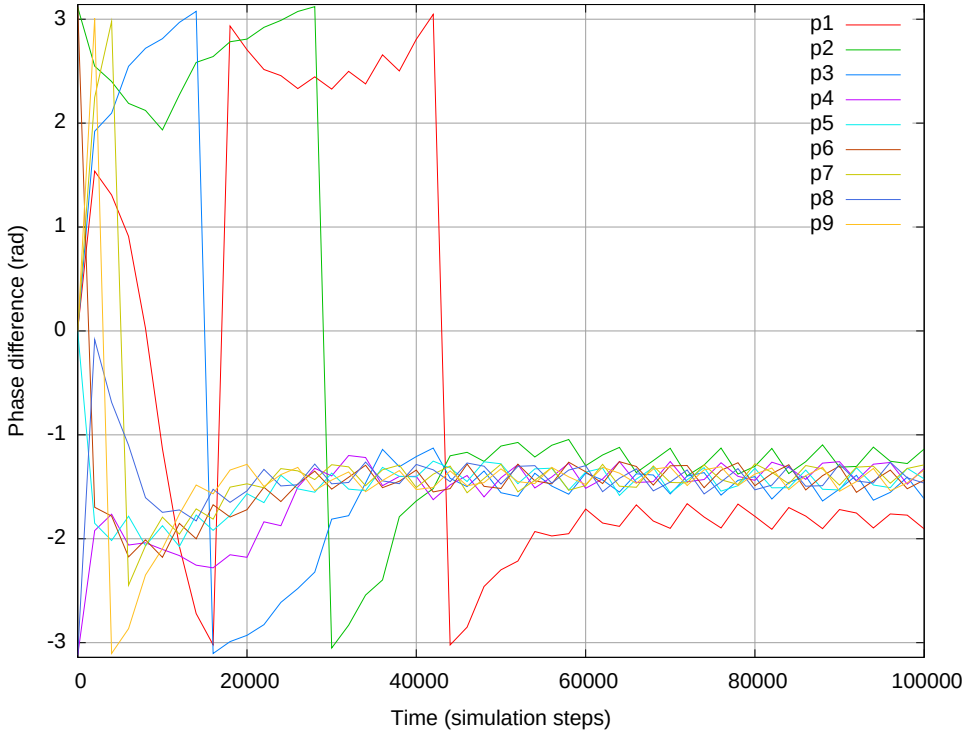


Figure 7.11: Time evolution of phase lags between modules with an ascending coupling strength of 2.0 and a descending coupling strength of 4.0. After an unstable transient period, modules autonomously find stable phase differences that guarantee a stable motor locomotion.

display a standing wave instead of a traveling one, a requisite for effective locomotion.

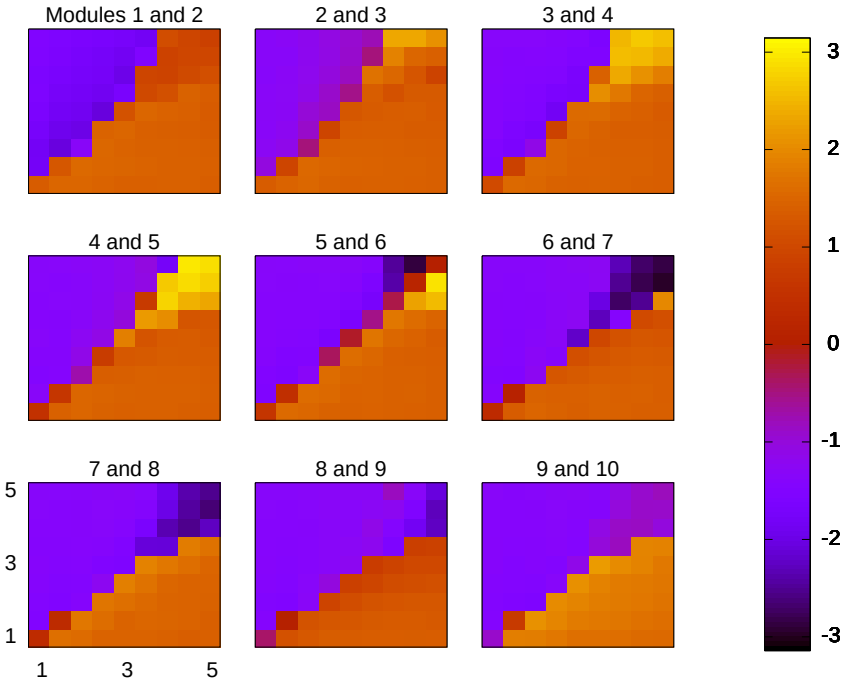


Figure 7.12: Pairwise phase lags between adjacent modules for different ascending and descending coupling strength. For each pair of couplings we have simulated our controller starting from arbitrary, not necessarily synchronized conditions. Each panel shows the mean phase difference between two adjacent modules for one run. Each simulation takes 4 000 000 steps; motor signals have a mean period of approximately 1400 points. As predicted by the theory, if one direction has stronger inhibition it will dominate and define the direction of travel. As observed in all panels, stronger ascending coupling induces a positive phase difference while descending coupling induces a negative phase difference. Symmetric coupling not always converges to a stable rhythm so the values shown here may vary from simulation to simulation.

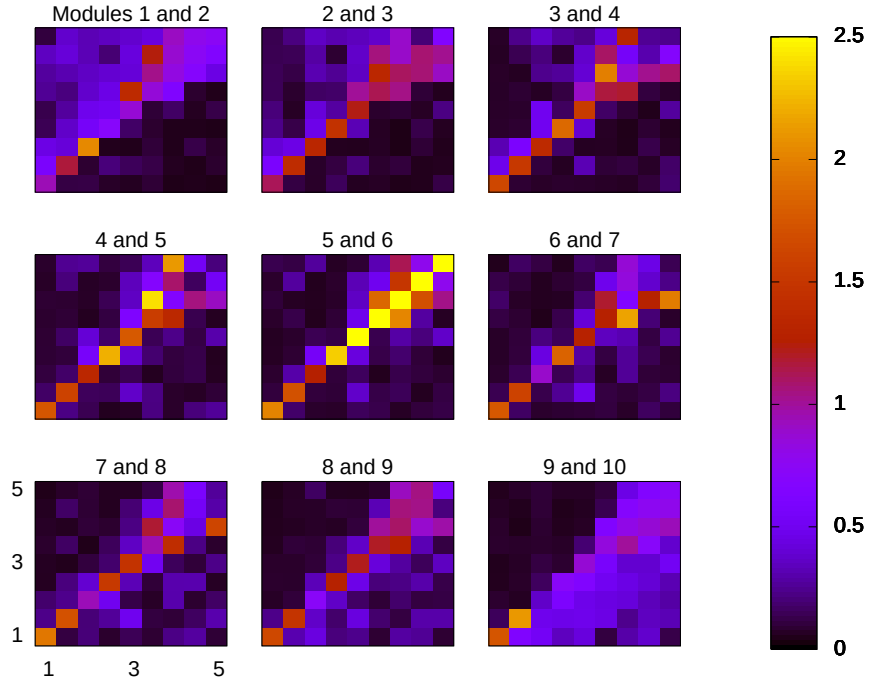


Figure 7.13: Standard deviation of phase lags between adjacent modules when varying coupling strength between them. We have run several simulations changing the ascending and descending coupling strength. For each pair of couplings we have simulated our controller starting from arbitrary, not necessarily synchronized conditions. Each panel shows the standard deviation of phase difference between two adjacent modules for one run. That is, higher deviation from the mean phase lag implies lower levels of synchronization, possibly down to no stable rhythm at all. Each simulation takes 4 000 000 steps; motor signals have a mean period of approximately 1400 points. Clearly asymmetric (one dominant direction) coupling yields very stable rhythms while symmetric coupling hardly ever converges to a stable rhythm.

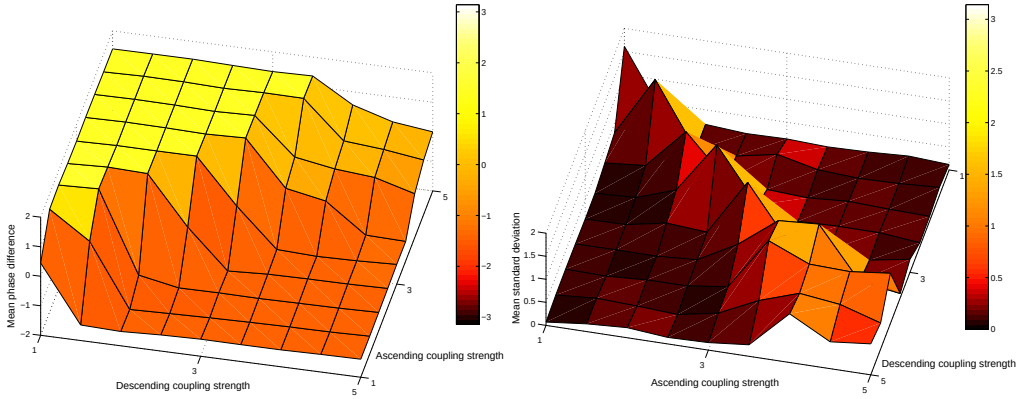


Figure 7.14: Left panel: mean of all panels shown in figure 7.12. Right panel: mean of all panels shown in figure 7.13. Our CPG effectively self organizes and achieves a state in which stable output motor signals with approximately constant phase difference are generated in the case of asymmetric coupling. In the case of symmetric coupling the CPG is mainly unstable and modules are unable to keep a constant phase difference with their neighbors.

7.5 Inhibitory loop

It is clear that the previous symmetric approaches are not in the right direction. Weak coupling lacks a mechanism that imposes a traveling direction and strong coupling is not able to generate a traveling wave at all.

This scheme is the first asymmetric connectivity proposed. The goal is to achieve a constant delay between adjacent modules by adding a specific circuit that creates a constant phase difference.

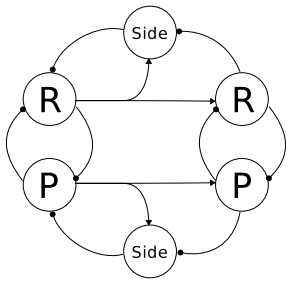


Figure 7.15: Inhibitory loop (remotors side symmetric, omitted for clarity).

As figure 7.15 shows, there is an intermediate neuron labeled "side" between every two promotor and every two remotor neurons. This neuron is in a silence regime except when excited, in which case it enters a tonic spiking regime. When the excitation ceases, "side" neurons will return to their silent state.

Parameters of the examples are: $\alpha = 15$, $\mu = 0.001$, $\sigma = -0.33$, $\beta = 0$ and $\sigma_e = 1$ for promotor and remotor neurons; $\alpha = 3$, $\mu = 0.001$, $\sigma = 0.25$, $\beta = 0$ and $\sigma_e = 1$ for "Side" neurons (silent when idle, tonic spiking when excited); $\lambda = 0.5$, $\beta = 10$, $\tau = 0$, $E_{syn} = 13$, $g_{syn} = 1.5$, $T = 1$ and $release\_time = 0.01$ for contralateral synapses; $\lambda = 0.5$, $\beta = 10$, $\tau = 0$, $E_{syn} = -7$, $g_{syn} = 3$, $T = 1$ and $release\_time = 0.01$ for excitatory synapse to neighbor neuron; $\lambda = 0.5$, $\beta = 10$, $\tau = 0$, $E_{syn} = -2$, $g_{syn} = 5$, $T = 1$ and $release\_time = 0.01$ for excitatory synapse to "side" neuron; $\lambda = 0.5$, $\beta = 10$, $\tau = 0$, $E_{syn} = 1.1$, $g_{syn} = 5$, $T = 1$ and $release\_time = 0.01$ for inhibitory synapse to "side" neuron; $\lambda = 0.5$, $\beta = 10$, $\tau = 0$, $E_{syn} = 9$, $g_{syn} = 5$, $T = 1$ and $release\_time = 0.01$ for inhibitory synapse from "side" neuron to motor neuron.

Discussion

The system works as follows. Let's take only one P neuron and its "side" neuron (the one it excites and receives inhibition from). The P neuron excites the "side" neuron when it (P) bursts. Under excitation, the "side" neuron enters a tonic spiking regime, which activates the inhibitory synapse to P. This inhibition is weak, so it does not completely inhibit P, but it reduces its frequency. The "side" neuron will remain in the tonic spiking regime for some time, even past P's burst, until it eventually returns to the silent regime. At a steady state, P's frequency under "side"'s influence is lower than its nominal frequency.

We now introduce one neighbor P neuron, let's call it P', to which P excites and who, in turn, inhibits "side". P' at its nominal frequency oscillates faster than P with "side"'s influence. If P' receives excitation it will oscillate even faster. Furthermore, at equal frequency of oscillation, excitation will tend to synchronize both P and P' in phase.

Let's see how this subsystem with two promotor neurons and one side neuron works. P' is oscillating faster than P, so it will slowly reduce its phase difference with the latter. By reducing phase difference it inhibits "side" earlier every cycle, so its spiking episodes become shorter. With less inhibition in each cycle, P begins to oscillate faster.

There is a given phase difference between P and P' that is a stable state. That means that both P and P' oscillate at the same frequency, which seems surprising given that P only receives inhibition. The fact is that at this point, inhibition from side only affects burst length, making it shorter, but does not affect P's recovery process, thus effectively making frequency higher than its nominal frequency. Here, "side" acts as a phase detector.

From the phase differences (figure 7.16) we can tell that this model achieves a very stable rhythm with stable phase differences between all modules. Burst periods are relatively regular, with border modules having less variation than middle ones.

To investigate the effect of the loop, we first suppress it and leave only the one-way excitatory path. Figure 7.17 clearly shows that phase maintenance can be achieved with only the excitatory path.

It is known that excitatory activity promotes in-phase synchronization of neurons. In this case phase difference is non-zero due to the delaying effect of the synapses. Nevertheless, a one-way path is not robust to perturbations, since information of what happened cannot reach all modules, only those located after the affected module on the path.

We now take the full CPG again and increase the conductance of the inhibitory synapse from the "side" neuron to the promotor neuron from $g = 5$ to $g = 7$. Figure 7.18 shows the effect of a stronger inhibition.

There are two different regimes during the evolution of the system. A first regime of approximately the same phase differences, most probably caused by a rapid excitatory synchronization, and a second regime of much higher phase differences. The shift of regime is propagated from p10 to p1, in the sense that it happens in p10 and then successively in all neurons down to p1. This could be a slower phase adjustment caused by the inhibitory loop. Periods become larger in this second regime, so it is likely that it is inhibition who is causing the effect.

Higher values of the inhibitory synapse display this two regimes behavior, but activity is much more irregular.

In conclusion, this CPG achieves a stable oscillatory state with constant phase difference between modules. Although this could be achieved with only the one-way excitatory path, the inhibitory loop acts as a redundant phase detector that theoretically could help the robot recover better from perturbations. Furthermore, it has a benefit over the previous symmetrical CPGs in that the direction of displacement is controlled. As a disadvantage, in order to make the robot travel in the opposite direction we would need a replicated set of connections in the opposite direction, each circuit being activated depending on the direction of travel.

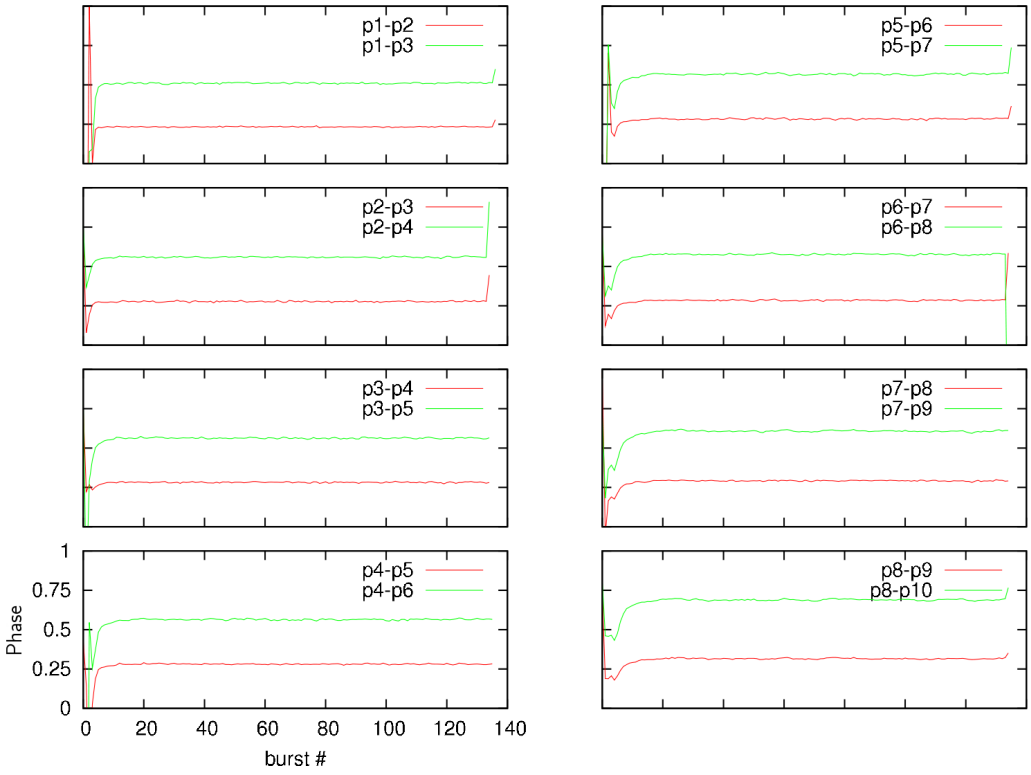


Figure 7.16: Phase differences for the basic inhibitory loop CPG. Phase differences are very regular across modules. Border neurons p1 and p10 show a slightly different phase, but not important for the generation of an efficient rhythm. There is a short transient of adjustment at the beginning. This small interval has a small local minimum, as can be observed in p7-p8. This is probably caused by the inhibitory circuit being activated.

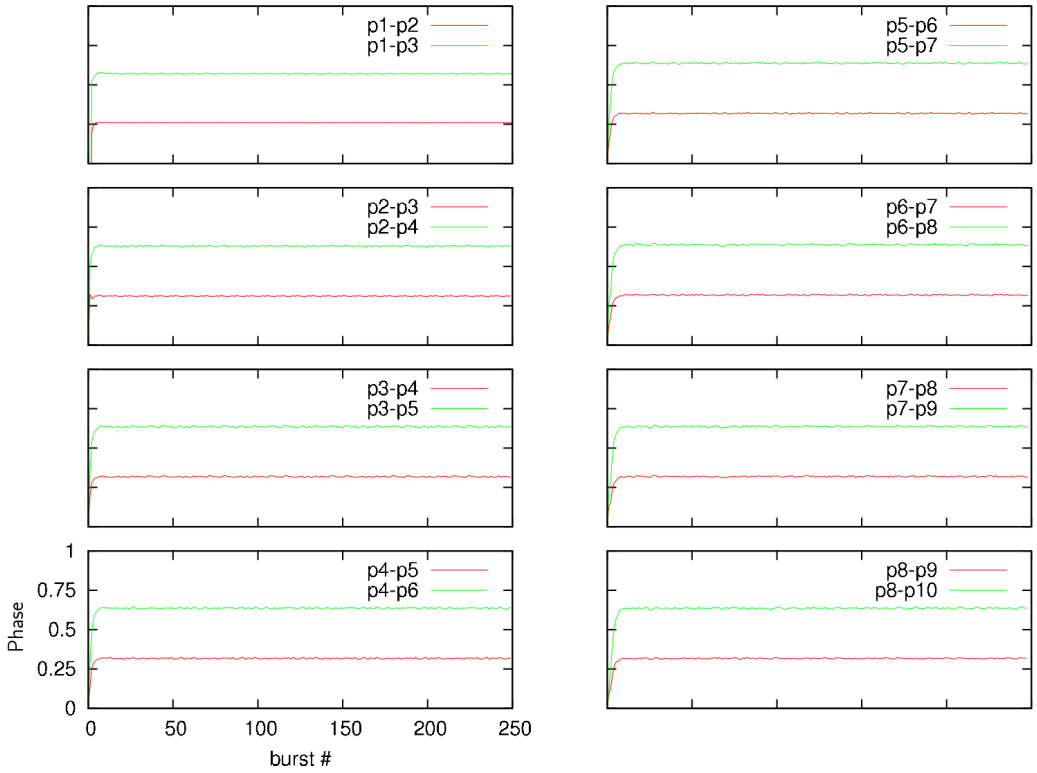


Figure 7.17: Phase differences for the inhibitory loop CPG, with the inhibitory path suppressed, excitatory path only. Phase differences are larger than in the closed loop case. The initial transient interval does not show the local minimum of the closed loop CPG.

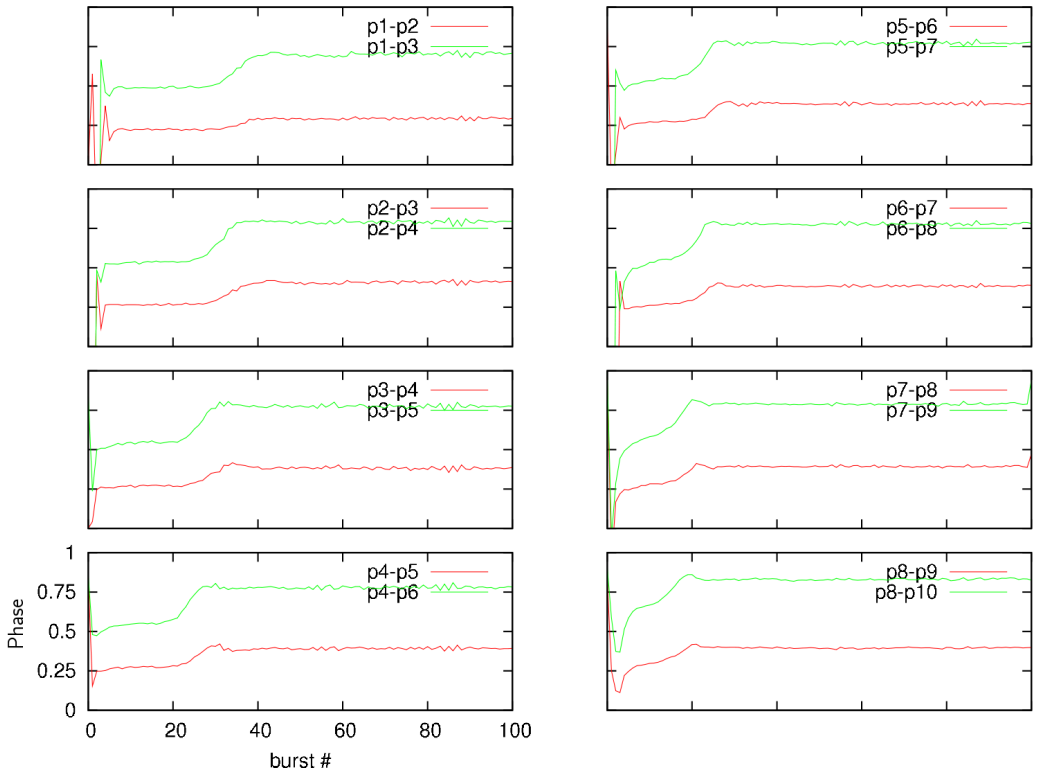


Figure 7.18: Phase differences for the inhibitory loop CPG, stronger inhibition from "side" to promotor neurons: $g = 7$. The beginning transient interval is longer than in the two previous cases. During the first transient the last modules slowly converge to their final stable state (p8-p9, p8-p10), but earlier modules show a stabilized behavior (p1-p2, p1-p3). As the last modules increase their phase difference, the loops propagate the effect backwards. This fact is reflected in the onset time of the second regime, with p1-p2 showing this effect almost 20 bursts later than p8-p9.

7.6 Push-pull model

Many circuits across the central nervous system show simultaneous excitation and inhibition. Contrary to what it may seem, this is not equivalent to a weaker excitation or inhibition (Kristan, 2007). We propose a connectivity pattern that takes advantage of this mechanism to implement a phase lag detection mechanism. The pattern is shown in figure 7.19. Figure 7.20 shows the behavior of this scheme.

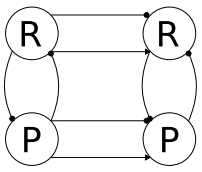


Figure 7.19: Push-pull model.

Parameters used in the simulations: $\alpha = 15$, $\mu = 0.001$, $\sigma = -0.33$, $\beta = 0$ and $\sigma_e = 1$ for neurons; $\lambda = 0.5$,$\beta = 10$, $\tau = 0$, $E_{syn} = -7$, $g_{syn} = .75$, $T = 1$ and $release\_time = 0.01$ for excitatory synapses; $\lambda = 0.5$,$\beta = 10$, $\tau = 0$, $E_{syn} = 8.5$, $g_{syn} = 1.75$, $T = 1$ and $release\_time = 0.01$ for ipsilateral inhibitory synapses.

7.6.1 Discussion

Synaptic input from one module to its neighbor depends on phase difference. Both excitatory and inhibitory synapses have the same time constants, the only difference being their reversal potential. Recalling equation 5.8:

$$I(t) = g \cdot r(t) \cdot (X_{post}(t) - E_{syn})$$

The combined result of both will depend on whether the postsynaptic neuron's potential is closer to the excitatory synapse's E_{syn} or to the inhibitory's. That is, if the postsynaptic neuron is hyperpolarized (at a very negative value) when synapses are active, the end result will be inhibitory (membrane potential will be further from the inhibitory synapse's reversal potential, $E_{syn} = 8.5$ and will contribute more than the excitatory's $E_{syn} = -7$). On the other hand, if it is depolarized (bursting or spiking), its membrane potential will be closer to the inhibitory's reversal potential, thus excitation will contribute more.

So for modules oscillating at the same frequency there is an unstable equilibrium point when they oscillate completely in phase. Then, there will only be excitation. However, inhibition will appear whenever in phase synchronization is not perfect. A stable equilibrium state exists that maintains a constant phase difference, as shown in figure 7.20.

Let's analyze figure 7.20. The oscillatory behavior of the second module can be explained as follows. As already explained, the nature of the input from the first module to the second will depend on their phase difference: it will sometimes be excitatory and sometimes inhibitory. Inhibition decreases

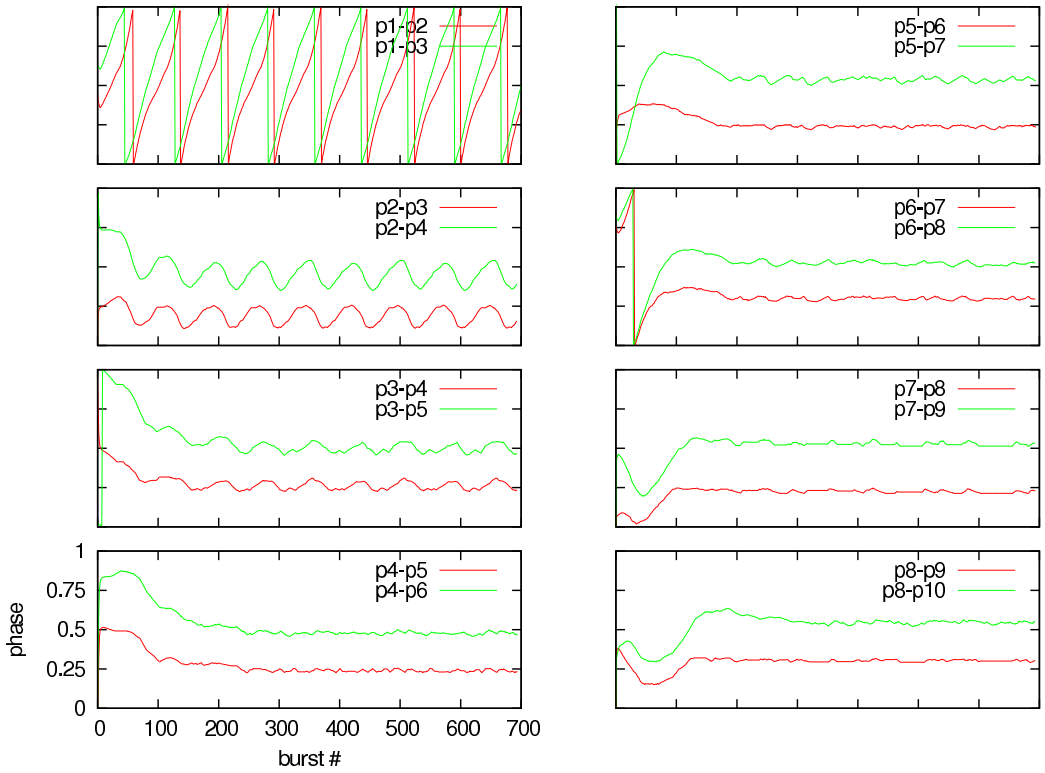


Figure 7.20: Phase differences for the push-pull model. Modules 2 and 3 oscillate at lower frequencies than module 1 because of their inhibitory input. There is no mechanisms that can adjust module 1's frequency, causing the phase drift shown in p1-p2 and p1-p3. Later modules receive input from already slowed down modules, and the CPG is able to maintain stable phase difference (p8-p9 and p8-p10).

the second module's frequency causing a phase drift between it and the first one. As they approach in-phase bursting, excitation restores the second module's frequency, but the first module still oscillates faster. Once the first module has overtaken the second one, inhibition reappears and slows down the second module again.

As figure 7.20 shows, this effect is less acute in later modules. It is because later modules receive input from an already slowed down module, and are able to slow down its successor too. This way, the CPG can reach a stable phase difference.

The key problem here is that there is no mechanism to adjust the first module's frequency to the slower oscillation of its followers. Thus, the need for a non-open (Huerta et al., 2001) topology becomes apparent. The connectivity, however, has proved to be an effective phase maintenance mechanism, given the two adjacent modules oscillate at a similar frequency.

7.7 Bistable intermediate neurons

The previously mentioned phenomenon of sequence inversion motivated this organization.

Let's assume that constant phase difference between adjacent modules is not as important as their keeping the right sequence. In order to achieve this, let's see what makes neurons keep a sequence.

In figure 7.21, two neural activities are shown: p3 and p5, synchronized in anti-phase. Rectangles indicate activity of a neuron, and the intervals labeled "1" and "2" indicate silent episodes between activities. These labels correspond to the two possible types of intervals in which a neuron could fire in order to make a stable sequence. The point is that a third neuron playing a role in the sequence can only fire in *one* of the two intervals, namely only after p3 and before p5 or vice versa. This is shown in figure 7.22.

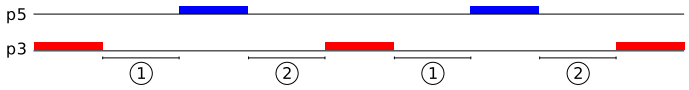


Figure 7.21: A sequence of two neurons, p3 and p5 oscillating in anti-phase and free intervals (1) and (2) where p4 can fire.

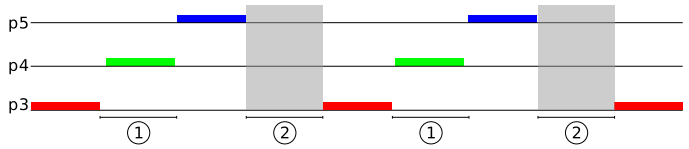


Figure 7.22: In order to achieve a stable sequence, p4 may only fire in one of the intervals (1) or (2). If it fires in (1), then it can only fire *after* p3 has fired and *before* p5 has done it; forbidden intervals for p4 marked gray

To design a mechanism that prevents a neuron from firing in forbidden intervals, we need a system that is context aware, in the sense that it has to be able to discern what type of interval it is in. This system will strongly inhibit the middle neuron (p4) in forbidden intervals and do nothing in allowed intervals.

Figure 7.23 shows a scheme in which intermediate neurons "B" have been added. These neurons can be in two states: tonic firing or silent. In which state the neuron is depends only on the origin of the last input it has received. That is, if the last input was excitatory, the neuron will be in the tonic spiking regime, *even if the excitatory input has ceased.* Likewise, if the last input was inhibitory, the neuron will be silent, regardless of whether inhibition is still in effect or not.

To be more clear, let's take one "B" neuron and call those neurons from which it receives input *right* and *left*, and let's call *middle* the neuron it inhibits. When *right* bursts, "B" enters the silent mode, meaning the *middle* neuron is allowed to burst between *right* and *left*. When *left* bursts, "B" quickly switches to tonic firing mode, totally inhibiting *middle*'s activity. This way, *middle* has only one chance to burst per cycle.

Bistable neurons are based on the Rulkov map with some modifications. In order to make the neuron bistable, parameter σ is manipulated. Two models have been implemented: in the first case σ is instantaneously changed according to the sign of the input; in the second case, σ is treated as a variable, σ_n, with a governing equation with two stable equilibrium points. This equation makes σ_n switch from one state to the other depending on input, but with a smoother non-instantaneous transition.

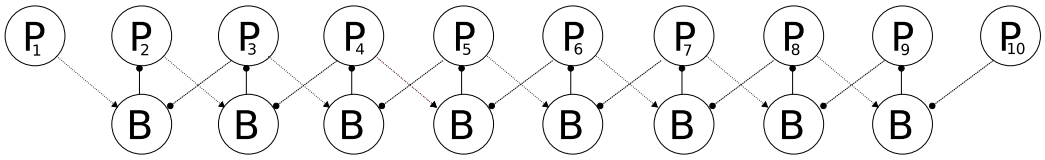


Figure 7.23: Bistable neurons model (remotors side symmetric, omitted for clarity). Bistable neurons 'B' are either in tonic spiking regime or silent. An excitatory input will put them in tonic spiking. An inhibitory input will drive them to their silent state. When either input ceases, the neuron will remain in the same state until a new input arrives. When in tonic firing, they will completely inhibit their corresponding 'P' neuron. Note the lack of direct interaction between the promotor neurons.

7.7.1 The bistable neuron model with instantaneous state transition

The bistable neuron model is based on Rulkov's. The neuron is made to oscillate between two states of silent and tonic firing by changing parameter σ depending on synaptic input. The complete equations are 7.1 through 7.4.

$$f(x,y) = \begin{cases} \frac{\alpha}{1-x} + y & \text{if } x \leq 0 \\ \alpha + y & \text{if } 0 \leq x < \alpha + y \\ -1 & \text{otherwise} \end{cases} \tag{7.1}$$

$$x_{n+1} = f(x_n, y_n) \tag{7.2}$$

$$y_{n+1} = y_n - \mu(x_n + 1) + \mu\sigma_n \tag{7.3}$$

$$\sigma_n = \begin{cases} 0.33 & \text{if } I_n > 0 \\ -0.33 & \text{otherwise} \end{cases} \tag{7.4}$$

Due to the opposing nature of the input to bistable neurons (one inhibitory and one excitatory input), the neuron will change its state depending on which presynaptic neuron has fired last. The change in this case will take effect immediately, since σ is changed as soon as I_n changes its sign.

Figure 7.24 shows the evolution of relative phases of the bistable neurons model in figure 7.23 with the immediate bistable neuron model. Modules 2 to 6 have a relatively stable phase difference. There is a propagating wave along them, but it is not followed by the other modules. Modules 5 to 10 show *parity synchronization*. It can be observed how phase differences with second neighbors p6-p8, p7-p9 and p8-p10 are around 0 (or equivalently 1), and phase differences with first neighbors p6-p7, p7-p8, p8-p9 oscillates at around 1/2, which corresponds to parity synchronization mode 2 described earlier.

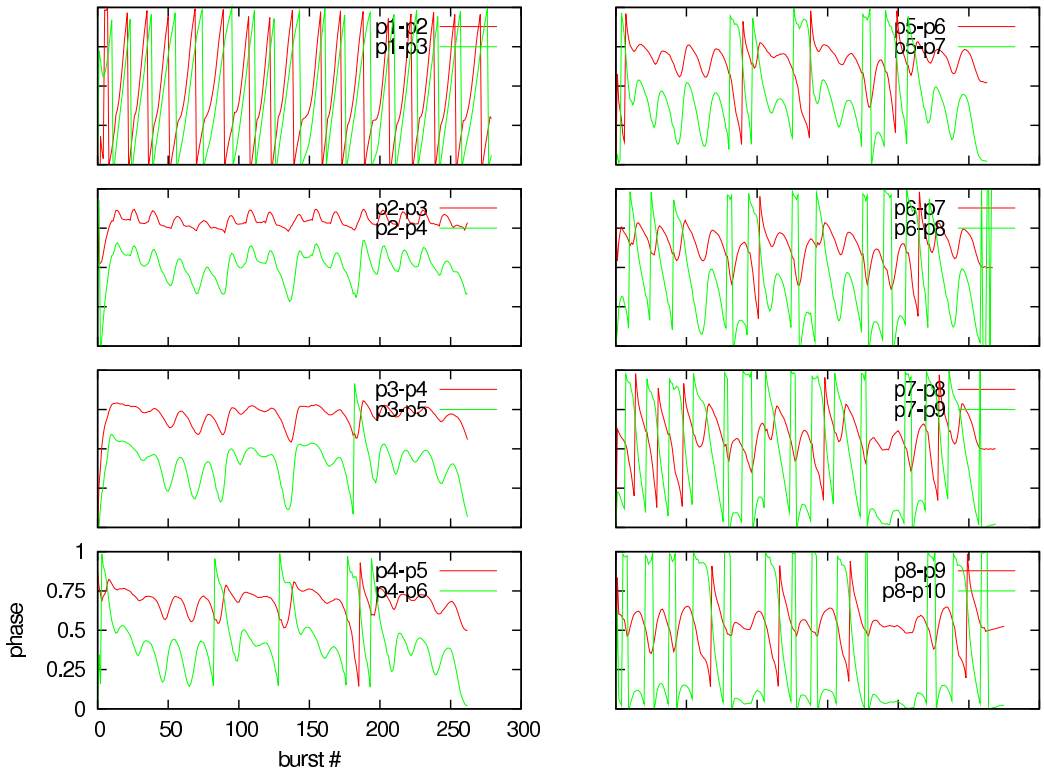


Figure 7.24: Phase differences for the bistable neurons model with instantaneous state transition. P1 receives no input, oscillating at its nominal frequency, higher than P2 and P3. Neurons P6 to P10 show a fuzzy kind of synchronization that we term "parity synchronization mode 2", with first neighbors synchronized approximately in anti-phase.

7.7.2 The bistable neuron model with delayed state transition

Instead of having σ sharply change from one value to the other depending on the sign of the input current, we now substitute this threshold in equation 7.4 with a map equation in 7.8. The complete equations are 7.5 through 7.8.

Now σ_n has a governing equation (7.8) with two stable equilibrium points. The role of I_n in this equation is to raise or lower the curve so that one of the stable states disappears and σ_n is forced onto the other stable state.

This dynamical approach displays a transitory period that introduces a delay in the response between the presynaptic neuron's activation and the bistable neuron's changing its state.

$$
\begin{aligned}
f(x,y) &= \begin{cases} \frac{\alpha}{1-x} + y & \text{if } x \leq 0 \\ \alpha + y & \text{if } 0 \leq x < \alpha + y \\ -1 & \text{otherwise} \end{cases} && (7.5)\\
x_{n+1} &= f(x_n, y_n) && (7.6)\\
y_{n+1} &= y_n - \mu(x_n + 1) + \mu\sigma_n && (7.7)\\
\sigma_{n+1} &= p \cdot (\tanh(2\sigma_n) + I_n) && (7.8)
\end{aligned}
$$

As observed in figure 7.25, phase differences are now much more stable. There is, however, an oscillatory behavior and some irregularities at the border modules. This is because of the lack of feedback from the rest of the body. Since all modules are somehow inhibited except for the border modules, their frequency is higher than the rest, causing a drift in their relative phases that causes periodic modifications of the overall rhythm.

7.7.3 Counteracting border effects

To counteract border effects we add a single synapse from two interior neurons to border neurons. This should have the effect of coordinating border activation rhythms by making the topology *non-open* (Huerta et al., 2001) and thereby adjusting the borders' frequencies. The diagram is shown in figure 7.26. The synapses depart exactly from the borders' second neighbors, so that these two neurons define a context for the border modules' first neighbors.

Figure 7.27 reflect this stabilizing effect of making the topology *non-open*. Phase differences are however still not constant along the body, with module 10 displaying a slightly deviated phase difference.

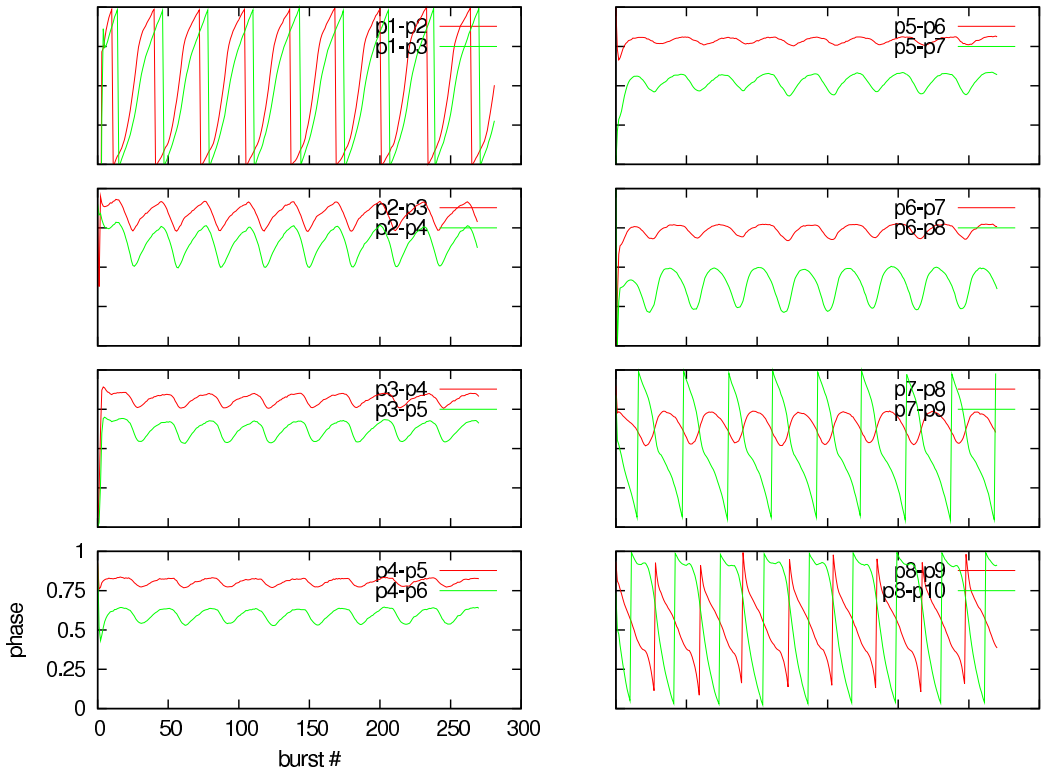


Figure 7.25: Phase differences for the bistable neurons model with delayed state transition. Activity is much smoother than with instantaneous transitions. Border effects still noticeable.

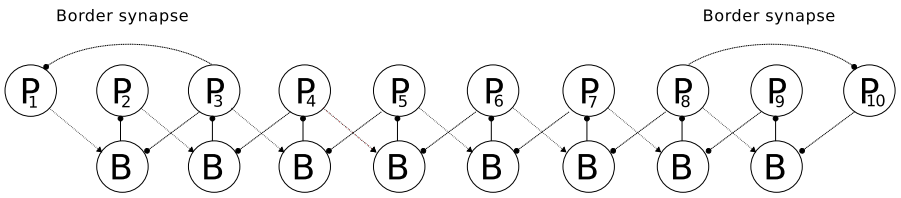


Figure 7.26: Bistable neurons with a *non-open* topology to cancel border effects caused by border neurons not having any input. (remotors side symmetric, omitted for clarity)

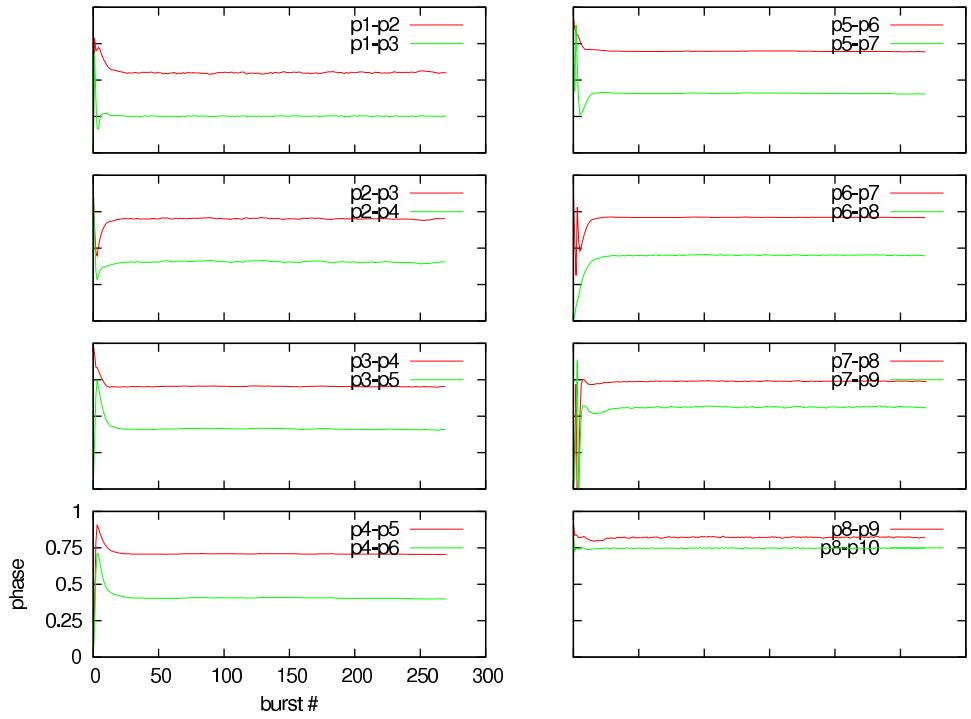


Figure 7.27: Phase differences for the bistable neurons model with delayed state transition and border effects compensation. P1 and P10 are now synchronized with the rest of the neurons. The synapse to P10 causes a slightly deviated behavior.

7.7.4 Evaluation

In conclusion, this model has proved to generate a stable firing sequence provided that border neurons receive feedback in order to adjust their frequencies.

The mechanism that coordinates the sequence does not operate through small frequency adjustments, like the previous CPGs, but through total inhibition of neurons. By letting promotors fire only on allowed intervals the direction of travel is established and maintained across modules. Side neurons with delayed state transitions are responsible for maintaining an efficient phase difference.

7.8 Summary

From recent research on living CPGs (see introduction of this thesis) we know several mechanisms that contribute to the richness of properties of these circuits. One of the most remarkable aspects about biological CPGs is their adaptability on various levels. On a high, evolutionary level, different animals, and even different species, share similar anatomical structures and their controllers. This encompasses a strong variability from animal to animal, yet evolution has succeeded in adapting to the particulars of each species. On a temporal, long term level, a single individual is faced with changes throughout its entire life. Different strategies help CPGs grow to successfully accommodate changes in size of their host. Finally, on the short-term, rich intrinsic dynamics of their elements and the right connectivity, particularly non-open topologies, provide for attractive behavior and fast transient dynamics. This ensures that CPGs will produce stable rhythms and that perturbations will be handled robustly.

Despite all this vast variability, there remains an intrinsic, unchanging property of CPGs that makes it possible for swimming animals to *swim*, to digestive systems to *digest* and so on. There is something essential to those activities, irrespective of the actual details of their execution. There is a *dynamical invariant* in each one of those tasks that must be preserved.

In this chapter we have contributed a collection of connectivity patterns for modular CPGs. They help implementing sequence keeping CPGs, to a long extent independently of the actual parameters of the implementation. The examples included in this thesis are presented with concrete parameter values as an illustration. However, the topologies are general enough to implement the same dynamical invariants with different parameter values.

The first proposed CPG was the "symmetric inhibition". The idea is very simple: connect the neurons with inhibitory synapses to prevent in-phase bursting and see if they can burst in a stable sequence.

This circuit adjusts itself by performing small corrections in neurons' frequency. This way phase difference can be adjusted by drift, that is, one neuron oscillating slower than its neighbors and letting them "drift away". As a result of this slow divergence, there is a long transient interval until a stable state is reached.

Drift may cause "sequence inversion", when one neuron overtakes its neighbor, meaning that the direction of the firing sequence may be reversed. Since the proposed circuit lacks a mechanism to impose a particular direction of travel, the final direction once a stable state has been found is unpredictable.

This, however, means that, if a suitable mechanism to define the direction

of travel is developed, like setting more specific initial conditions rather than random ones, the same set of synapses could be used for both forward and backward travel. Another mechanism would be needed for "on the fly" change of direction, nevertheless.

The "symmetric inhibition" CPG is a good candidate if the goal is to maintain an already established firing sequence, in both directions. However, other mechanisms should be used to start the sequence.

Inhibitory loop This CPG works by correcting one module's frequency depending on its phase difference with its successor. Through a direct synapse from one module to its neighbor, adjustments can be made to the successor module and through the loop, adjustments are also made to the predecessor module.

The CPG is designed to work in a particular direction. The circuit is capable itself both of maintaining the direction of travel as well as of establishing it in the first place. The transient interval is relatively short, depending on synapse conductance. Real experience is needed to determine if the transient is tolerable or indeed too long.

The "inhibitory loop" CPG is a very stable CPG, capable of starting and maintaining a stable sequence. The loop enables bidirectional adjustment to successor and predecessor modules.

Push-pull. The "push-pull" CPG is a very simple circuit, that nevertheless achieves a very stable phase difference between modules oscillating at similar frequencies.

This CPG works as a phase detector. The net effect of the two synapses, one excitatory and one inhibitory, projected onto the successor module depend on the exact phase of the cycle on which the postsynaptic neuron finds itself. This way, modules can be sped up or slowed down until the desired phase difference is achieved.

The projected synapses go only one way. This means that information flow is unidirectional. Predecessor modules cannot process information coming from later modules.

One side effect of unidirectional information flow is that the first module oscillates faster than the rest. Since it does not receive any input, its frequency is unaffected, being it its nominal frequency. All other modules oscillate always at a lower frequency, so a stable phase difference (between the first modules) is impossible to achieve.

The "push-pull" CPG is able to achieve stable phase differences provided that adjacent modules oscillate at approximately the same frequency. As it is presented here, the lack of feedback to earlier modules prevents the adjustment of the first module's frequency to its successor's and causes periodic

variations in frequency of the second and third modules.

Bistable neurons. The "bistable neuron" CPGs work not through frequency adjustment, but through complete inhibition of neurons. The endogenous rhythmic nature of neurons provides for an intermittent inhibitory activity. Topology then drives the system in order to create a wave of successive inhibitions and activations, defining allowed intervals in which promotor and remotor neurons may fire. This mechanism defines a consistent direction of travel and prevents sequence inversion.

The effect of strong inhibition is to "erase" the past, setting the system in a given state independent of initial conditions. As a result, this CPG might be very resilient to perturbations.

Lack of feedback to border neurons again prevents complete synchronization. However, we provide a model with two synapses that counteract this effect. As shown, just by establishing these two synapses that correct the border modules' frequencies, the CPG is able to achieve a constant phase difference among all of its modules.

Reversibility is here considered a feature, but it must be noted that the only reversible CPG is also the only one that is not capable of defining a direction of travel.

The "bistable neurons" CPGs have bidirectional information flow indirectly. Promotor [remotor] neurons are not directly connected to each other, but one module can affect both of its neighbors through their bistable 'B' neurons.

7.8.1 Symmetry

In general, it could be said that symmetry is to be avoided. This is not to say that no symmetry should appear, rather that it must be broken at some level. If topology is symmetric, like the first symmetrical model proposed, then initial conditions must break the symmetry. This CPG has the basic ability of adjusting itself to a stable sequence but has no mechanism to determine the direction of movement, so both directions are equally likely given random initial conditions.

If activation of the modules is symmetric, that is, in-phase or anti-phase synchronized, there will not appear any propagating wave that makes the robot undulate, but rather there will be a standing wave. In conclusion, activations must be periodic but not symmetric, i.e., there must be a constant phase difference across modules. Otherwise the generated movement may not be efficient. *Parity synchronization* modes shown above are an example of symmetry to avoid.

7.8.2 Coupling strength

Weak coupling is a mechanism to exchange information among modules without *severely* altering the receiver's behavior. It allows each module to keep its activity seemingly unaffected and independent while at the same time coordination is being "negotiated". The disadvantage is that for large systems convergence can be very slow if initial states are sufficiently different.

On the other hand, strong inhibition can sharply change the receiver's activity. This mechanism allows for a faster convergence to a stable state, independent of initial conditions, and a faster response to changes. This is also a good mechanism to prevent against perturbations, since strong coupling has a resetting effect that *erases* the past, thus isolating perturbations in a small time window.

Probably the best approach is to combine both mechanisms and have different connectivities acting at different times: one with stronger coupling to drive the system to a known state (for starting and stopping the movement, or to recover from perturbations) and a weaker system to maintain activity once the stronger one has been shut off.

7.8.3 Coupling sign

While our first approach was to use inhibitory synapses only, this work has shown that excitatory coupling can also be useful.

Inhibitory synapses act to prevent in-phase synchronization and regulate oscillation frequency. In general, when one module receives an inhibitory current, its frequency is decreased. This is the cause for phase drift in the absence of feedback to the presynaptic neuron.

In some cases, however, inhibition may cause an increase in frequency. If the timing is right, inhibition may shorten a neurons' bursts but leave its recovery process unaffected, thereby shortening bursting period.

Excitatory synapses, on the other hand, promote in-phase synchronization. Taking advantage of the delayed effect of chemical synapses, excitatory coupling can be used to achieve a constant delay between coupled modules.

The push-pull model has shown that these types of coupling are not exclusive, nor are they equivalent to a weaker version of one of them when used simultaneously. Indeed, using both simultaneously we have constructed a stable phase detector.

7.8.4 Topology

It is very important that all modules share information, that is, send *and* receive stimuli. CPGs based on inhibition will suffer border effects if border neurons do not have any feedback. If this is the case, they will not have

any means to adapt their frequency of oscillation to that of the rest of the modules. As a consequence, the system will not be able to achieve stable phase differences among them.

We have shown that it is possible to coordinate modules on a global scale by using first neighbors only connections. This is however not very resistant to failure since malfunction in one module will completely break the chain of communication. Projections to further modules and redundant communications will certainly provide richer dynamics and better fault tolerance.

8 Robot and CPG integration

In the previous chapters we have introduced different bio-inspired building blocks to construct modular CPGs. First, a modular oscillator based on the half-center principle, and then different inter-module connectivity patterns. These fundamental elements and strategies are general mechanisms that can be implemented in different specific ways.

Combining these elements we can build effective modular robot controllers, that offer the flexibility, autonomy and robustness characteristics of living CPGs. In this chapter we illustrate these strategies with a concrete example. Combining the negotiation capacity of the modular oscillator introduced in chapter 6.1 with one of the dynamical preserving connectivity patterns presented in chapter 7, we will build a CPG to control a real modular robot. First, we will study the properties of the control signals using the analysis tools described in chapter 5.3. Then, we will study two representative examples of CPG activity: self-organization from arbitrary initial conditions and recovery from noise.

In this chapter we show the integration of the module oscillator, one of the proposed topologies and a real-world testbed robot with eight modules (González-gómez et al., 2006). We refer the reader to the introduction and figure 1.1 for an explanation of the architecture of the robot.

8.1 Case study: non-open topology

We will apply the analytic signals technique to analyze three parameters of the signals that command the motors of our robot, namely amplitude, frequency and phase difference among adjacent modules. We first study the behavior of the CPG depicted in figure 8.1.

Figure 8.2b shows the amplitude envelope of the signal generated by the bistable CPG in figure 8.1 for motor 8 of the robot. There is a transient period at the beginning of the simulation during which neurons P and R within the module are not yet synchronized. Amplitude of oscillation is low during this transient. Once the system stabilizes, oscillation reaches its nominal amplitude, which is kept constant for the rest of the simulation.

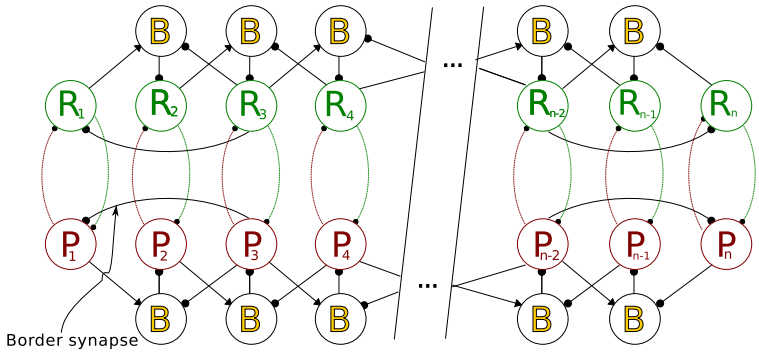


Figure 8.1: The "bistable" CPG. The architecture for each module is conserved. Inter-module coordination mechanisms are shown in thicker trace. Arrow terminated lines are excitatory synapses; ball terminated lines are inhibitory synapses. This CPG is based on the assumption that to achieve a stable firing sequence, each neuron may only fire in an allowed time window, defined by the bursting activity of both its neighbors. 'B' neurons are bistable neurons. When they receive excitation they enter a tonic spiking regime. If they receive inhibition, they will enter a silent regime. Under absence of input, they will remain in the same state they were. 'B' neurons effectively restrict the allowed intervals through a strong synapse. When a 'B' neuron is in its active state, it will completely inhibit one 'P' (respectively 'R') neuron until it goes back to the silent regime.

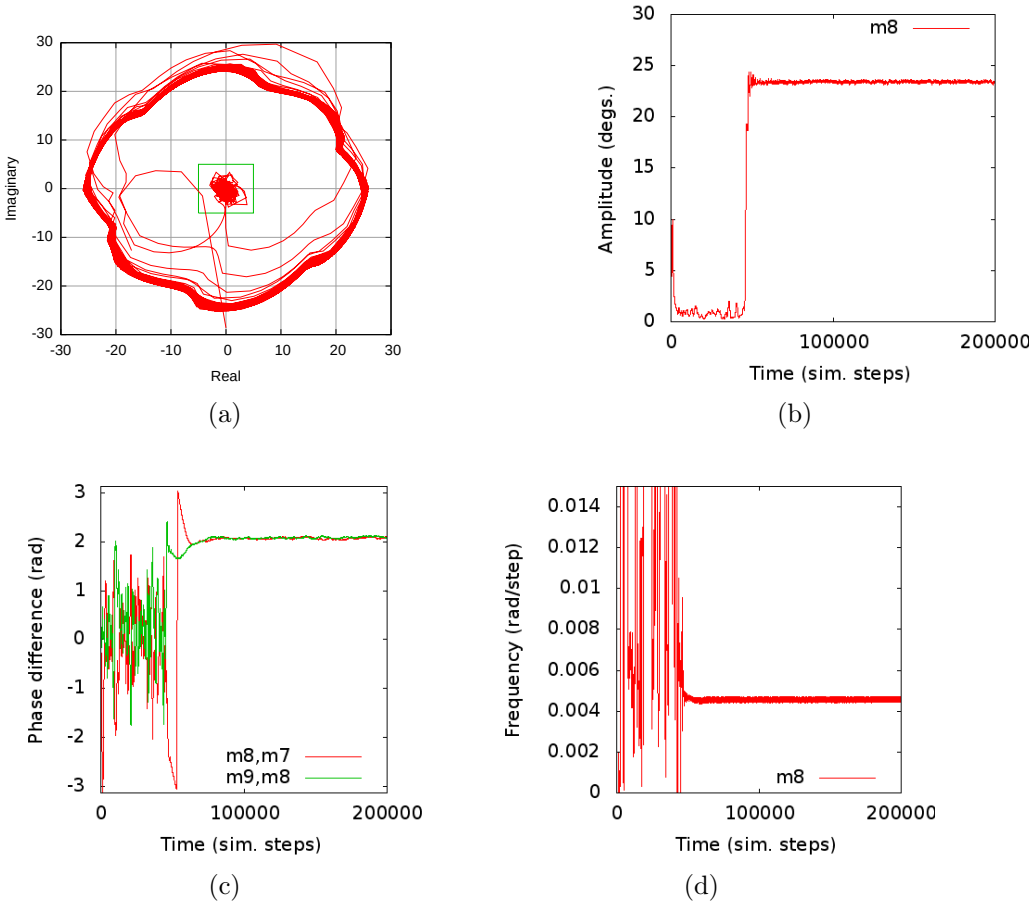


Figure 8.2: Analysis of the signal generated by the bistable CPG for module number 8. The system displays an initial transient period (marked with a green box in (a)) in which oscillations are not yet stable. After this, the system achieves a stable state in which amplitude, phase difference with neighbor modules and frequency remain constant. Signals generated for other modules show similar behavior, with possibly different transient periods but similar steady states. (a) Projection of the analytic signal generated by the CPG for module number 8. The shape is similar to the mono-component signals in figure 5.7. This gives an idea of the oscillatory behavior of the signal. The points in the center correspond to an initial transient period before the system achieves a stable state. (b) Amplitude envelope of the signal. (c) Phase differences between modules 8 and 7 and between modules 9 and 8. (d) Frequency of the signal.

Figure 8.2c shows the phase differences between modules adjacent to mod-

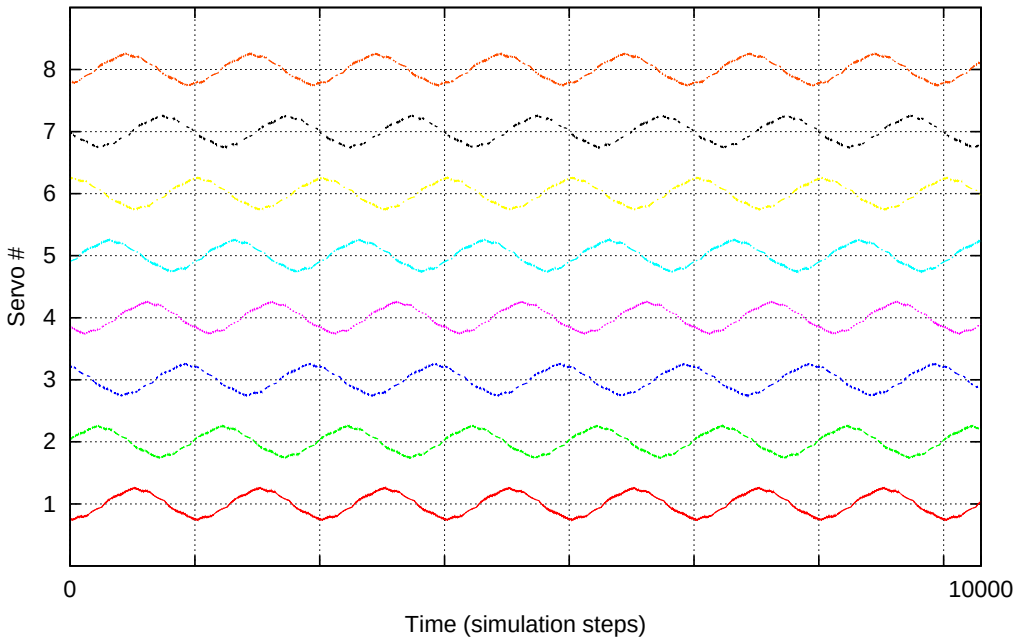


Figure 8.3: Snapshot of the CPG analyzed in figure 8.2. After an initial transient interval, the CPG runs synchronized and is working in stable regime. All signals show a steady oscillatory behavior with constant amplitude and frequency. Grid marks are set at intervals of 1370 time steps, approximately one period of oscillation. All signals maintain a constant phase relationship among them and elicit a stable locomotion.

ule number 8. There is a clear transient period at the beginning of the simulation in which modules are not synchronized. During this time, the system explores its state space trying to find a stable state. After the transient, phase difference between modules 8 and 7 and between 9 and 8 evolve in similar manners. At the steady state, phase differences are maintained constant. Other modules show similar behaviors.

Figure 8.2d displays frequency behavior of module 8 of this CPG. The mean frequency is kept constant during the whole simulation except for an initial transient. Intra-cycle frequency, however, is not constant. That means that the speed at which the signal changes is not constant within one cycle.

8.1.1 Case study: an open topology

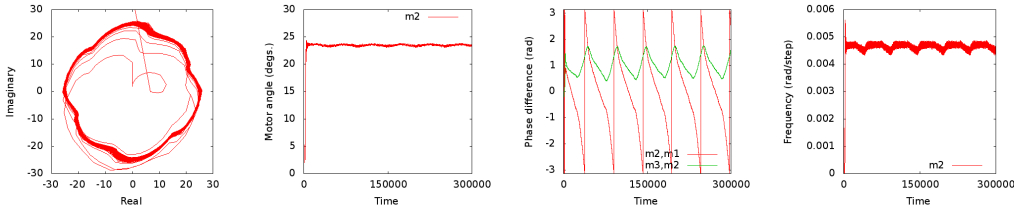
In the previous section we have addressed the analysis of a particular CPG built on bursting neurons, with strong inhibition and a non-open topology. The result is that all modules are capable of finding a stable oscillatory state, with constant amplitude and frequency, and with a constant phase difference between them.

We will repeat in this section the same analysis for an open topology. The basic structure is similar to that shown in figure 8.1, except that the border synapses have been removed. This way, border modules do not receive any input from other modules.

Figure 8.4a to figure 8.4d show the results of the analysis applied to this open topology. Border modules oscillate at a frequency slightly higher than the rest of the modules because there is no inhibitory synapse acting on them. This difference in frequency prevents synchronization between modules.Figure 8.4c clearly shows how phase difference between module 2 and module 1 (border module) drifts constantly. Inner modules 2 and 3 cannot synchronize, yet their phase differences diverge in a lesser degree. The result is that the CPG fails to generate an efficient locomotion in the robot.

The simulation procedure is as follows: the CPG software is run offline during 1 000 000 simulation steps. A program then reads each simulation step as a single line composed of the target positions for each motor. It selects one of every five lines so as to have an acceptable speed of locomotion, suitable for later video processing. Higher speeds can be achieved using a lower sampling rate, for instance one out of thirty simulation steps. Each target position is sent along with the number of the target motor over RS232, to a controller board (`http://www.iearobotics.com/proyectos/skypic/skypic.html`) running a program that generates a PWM signal that positions the motor at the specified angle.

We have carried out two tests to illustrate how the robot performs in the real world. The first one regards the stability of steady state locomotion of the



(a) (b) (c) (d)

Figure 8.4: Analysis of the signal generated for module number 2 by a CPG with an open topology. In this simulation the transient period is very short and the system quickly finds an oscillatory state. However, the system fails to maintain a stable rhythm and instead shows a metastable sequence. There are periodic variations in frequency that, in turn, prevent adjacent modules from keeping a constant phase difference. (a) Projection of the analytic signal generated by the CPG for module number 2. (b) Amplitude envelope of the signal. (c) Phase differences between modules 2 and 1 and between modules 3 and 2. There is a clear constant phase difference shift which prevents the necessary phase locking for efficient locomotion. (d) Frequency of the signal.

real robot. In the second one the robot is subject to a strong noisy perturbation, then its recovery is analyzed. Results are extracted from high-definition (hdv) video recordings (see figure 8.6) using a Sony HDR-HC9. Shooting was done from a distance of 2.2m, then videos were downloaded to a computer using dvgrab 3.5 on a linux machine. Uncompressed tiff frames where extracted from the video using ffmpeg, then converted to uncompressed JPG using ImageMagick. A video tracking software has been used on these frames to extract the positions of motor markers (figure 8.6).

8.1.2 Steady state locomotion

The first experiment consists of a free run simulation. We start the CPG with arbitrary initial conditions and let it evolve freely, with no perturbations. The output of the CPG is sent to the robot, which is recorded in video. Direct observation of the robot reveals that it performs an undulatory movement with steady forward locomotion. To illustrate this, we track the marker of one of the central modules and analyze the trace offline (see figure 8.7). In order to gain more insight about the performance of the robot, the signal is decomposed into its two coordinates with respect to time (figure 8.8). The vertical component of the module is periodic: this reveals that the movement of the marker is indeed oscillatory; due to the geometry of the robot the trajectory is not constant, with some plateaus of resting activity, but the

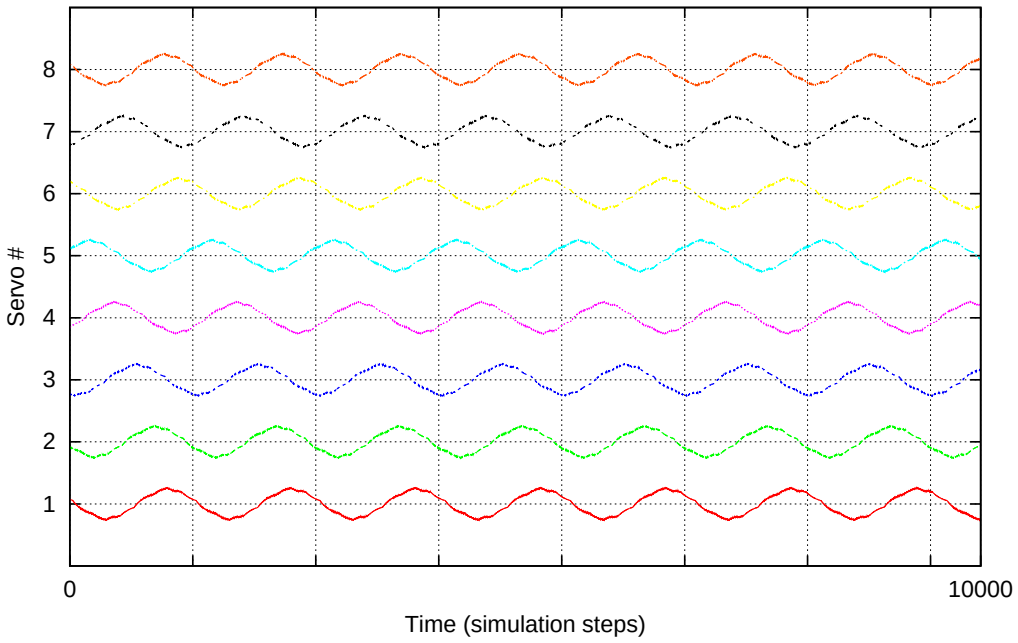


Figure 8.5: Snapshot of the CPG analyzed in figure 8.4. This CPG is not built with an open topology. That is, there are two modules in the topology that do not receive any feedback from the rest of the circuit. For this reason, the circuit is not able to maintain stable synchronization between signals, and their frequencies diverge periodically in time as shown in figure 8.4c. For this time slot, the three last signals show a markedly skewed phase in relationship to the other servos. Grid marks set at intervals of 1350 time steps, approximately the mean period of all signals. However, while signal number 2 has period that fits the mean, signal number 8 is clearly oscillating faster.

overall behavior is sinusoidal. Horizontal forward velocity is almost constant, with a mean value of 1.4cm/s approximately, again with small plateaus of quietude due to the robot's geometry.

This experiment confirms that our CPG can effectively drive the real robot and perform an steadily undulatory locomotion, with uniform forward velocity.

For comparison purposes we include figure 8.9 in which the analysis outlined in the previous section is performed on the vertical displacement of the marker shown in figure 8.7. This signal does not meet the requirements for the instantaneous phase to have a valid physical interpretation for every time instant (indeed, there is an artifact, surrounded by a rectangle, in figure 8.9 whose interpretation would be that the phase went backwards in time, which is physically impossible). However it is still valid to infer the mean behavior of the phase difference between adjacent modules with respect to time. Figure 8.9a shows the corresponding analytical signal, obtained using the Hilbert transform, projected on the complex plane. It is seen that the trace of the marker is oscillatory, stable and periodic. What is more, the module at hand is seen to keep a reasonably constant phase difference with its neighbors, as seen in panel 8.9b. Small oscillations in this panel are due to the fact that instantaneous frequency is not constant within one oscillation cycle. Effectively, coordination is guaranteed by the CPG, which can easily handle these small differences.

8.1.3 Recovery from noise

In the second experiment, simulation begins with the CPG at arbitrary initial conditions, as in the first one. The CPG is left to evolve and settle at a stable steady state. At a given interval in time, a noisy stimulus (random variable from a uniform distribution in the interval [0, 30)) is applied to all neurons in the CPG controlling the real robot. Analysis is performed from the onset of the perturbation to a point where locomotion is again stable (figure 8.10 and 8.11). During perturbation the robot lies flat on the ground without making any movement at all, not even small trembling of the motors. Once the perturbation is over, neurons resume their activity and a new synchronization process begins. During this process locomotion is ineffective and the robot undulates in-place, without traveling (see rectangle in figure 8.10). After a short time synchronization is achieved and an effective locomotion is re-established.

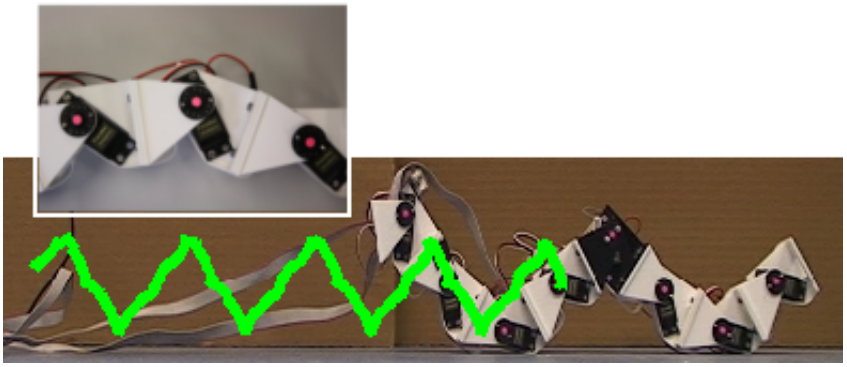


Figure 8.6: High definition motion tracking. Video was recorded using a Sony HDR-HC9 in high-definition (hdv), shooting from a distance of 2.2m, and downloaded to a computer using dvgrab 3.5 on a linux machine. Uncompressed tiff frames where extracted from the video using ffmpeg, then converted to uncompressed JPG using ImageMagick. The green trace corresponds to one of the middle segments of the robot. In the upper left corner the inset shows a close-up of the modules with the position markers used for locomotion tracking. Original footage is available as supplementary multimedia material (filename: stable.mpeg, 4.1MB, MPEG 1/2 video; filename: recovery.mpg, 7.0MB, MPEG 1/2 video).

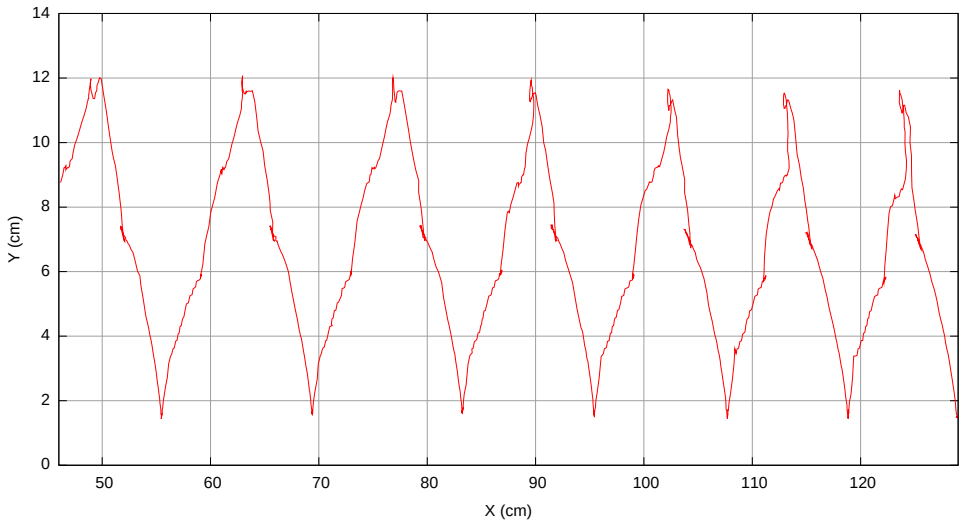


Figure 8.7: Video tracking of the middle segment of the real robot, locomoting in stable regime. X coordinate represents horizontal displacement from the left border of the recording area; Y coordinate represents vertical displacement from the ground. The trace of the point is clearly periodic, a consequence of the periodicity of the controller CPG.

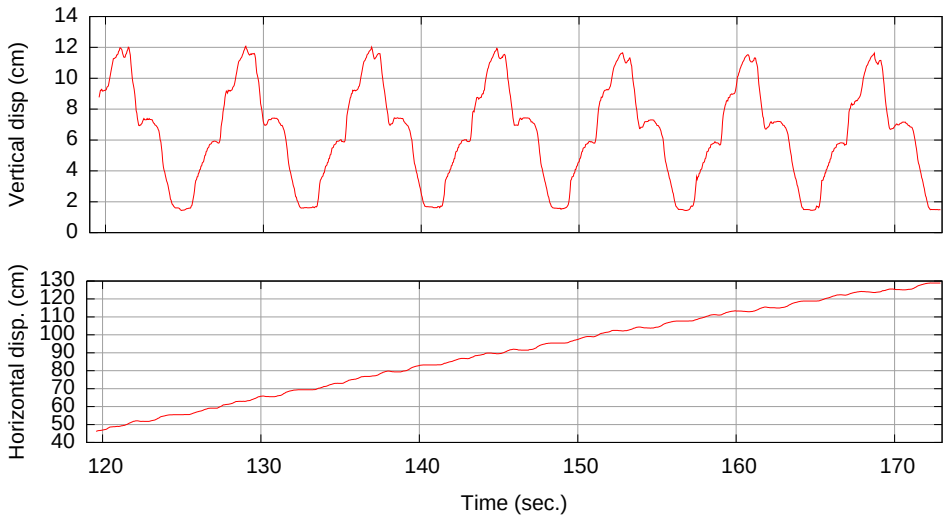


Figure 8.8: Video tracking of the real robot: vertical and horizontal coordinates with respect to time. The bottom panel shows a steady forward locomotion with an approximate overall speed of 1.4 cm/s.

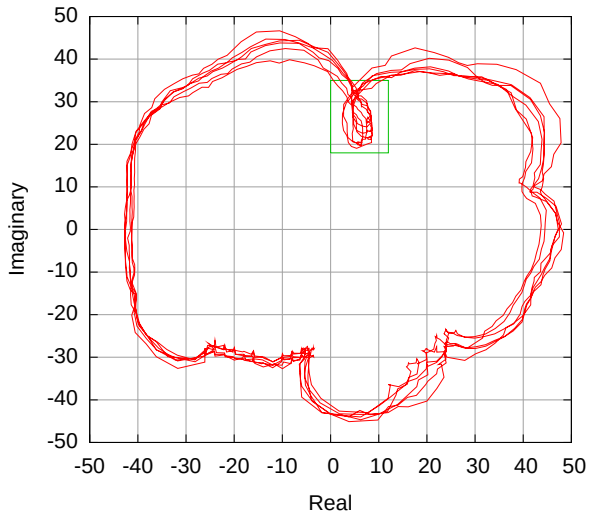
8.2 Further testing

We include some results of simulations performed using the other topologies proposed in this thesis. We present video tracking results with the asymmetric (section 7.4), the inhibitory loop (section 7.5) and bistable with a shorter transient.

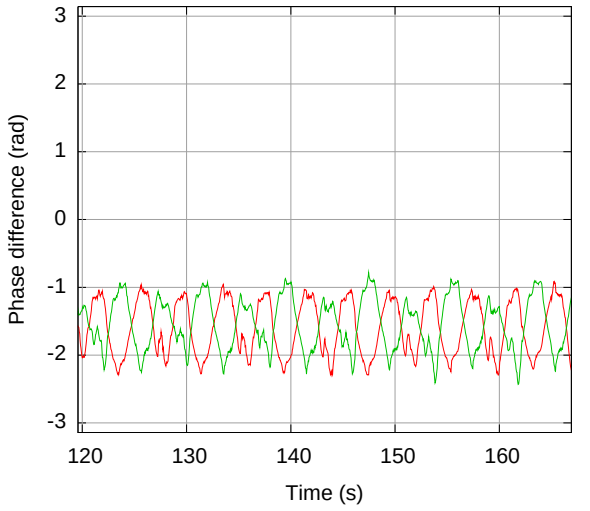
8.3 Summary

In previous chapters we have introduced generic building blocks for modular CPGs. In this chapter we integrate some of the proposed elements and build a CPG to control a real robot. We analyze two representative examples that showcase the properties of the CPG.

We show an example in which all neurons of the circuit are initialized with random initial conditions. The intrinsic dynamics of Rulkov's neuron model guarantees that there exists a stable limit cycle to which each neuron will converge. In this process, the selected inter-module topology enables global negotiation of the locomotion rhythm through local interactions. As a result, all units of the circuit collaborate, through adjustment of their activity, to establish a uniform, coordinated and robust locomotion. Our locomotion encoding mechanism and the motoneurons effectively generate smooth signals with an appropriate amplitude to control the real servos of the robot.



(a)



(b)

Figure 8.9: a) Hilbert transform of the vertical displacement of the central module of the real robot (panel a) in figure 8.8). The analyzed signal presents some anomalies (green rectangle) that make instantaneous frequency interpretation invalid (here the phase would go backwards, which is not a valid physical interpretation); however the analysis is still useful to convey an overall idea of the behavior of the system. The movement of the central module is rhythmically stable, with constant amplitude and frequency. b) Phase difference of the central module with its predecessor and successor modules. Since the analysis yields non constant instantaneous frequency, instantaneous phase difference oscillates in every cycle of the rhythm. However mean phase difference is clearly bounded (mean: $1.55 \approx \pi/2$ rad; std: 0.39 rad).

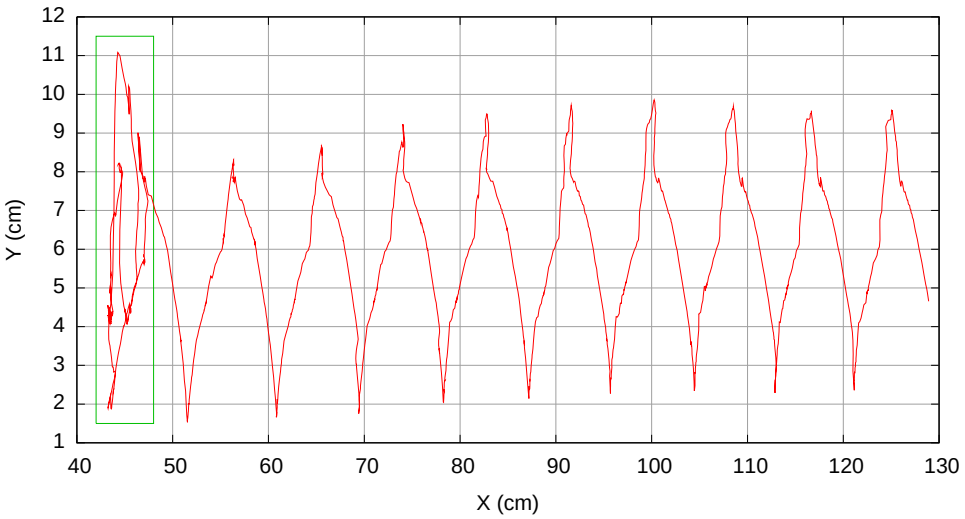


Figure 8.10: Recovery after a disruptive noise is applied to every neuron controlling the real robot. A normal simulation is carried out. At a given time a high level of noise (random uniform distribution in the interval $[0, 30)$) is injected into every neuron of the CPG. As a result of the noise, neurons stop displaying their normal bursting activity and the CPG generates no oscillations at all. Right after the noisy stimulus is released, neurons go back to their bursting behavior. The robot finds itself in an uncoordinated state (activity surrounded by a rectangle). Eventually, the robot resynchronizes itself and resumes forward locomotion.

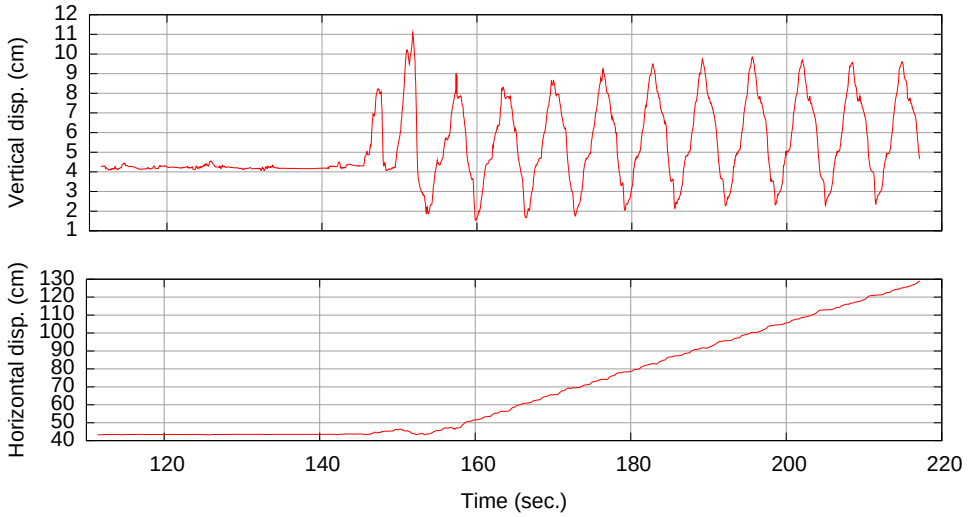


Figure 8.11: Recovery after a disruptive noise is applied to every neuron controlling the real robot. A noisy stimulus is applied that disrupts the activity of all neurons in the CPG. During this stimulus the robot lies still on the ground without making any move. After noise is removed, the robot begins searching for a synchronized state. After only two oscillations, the robot is again moving forward at an approximate speed of 1.5cm/s.

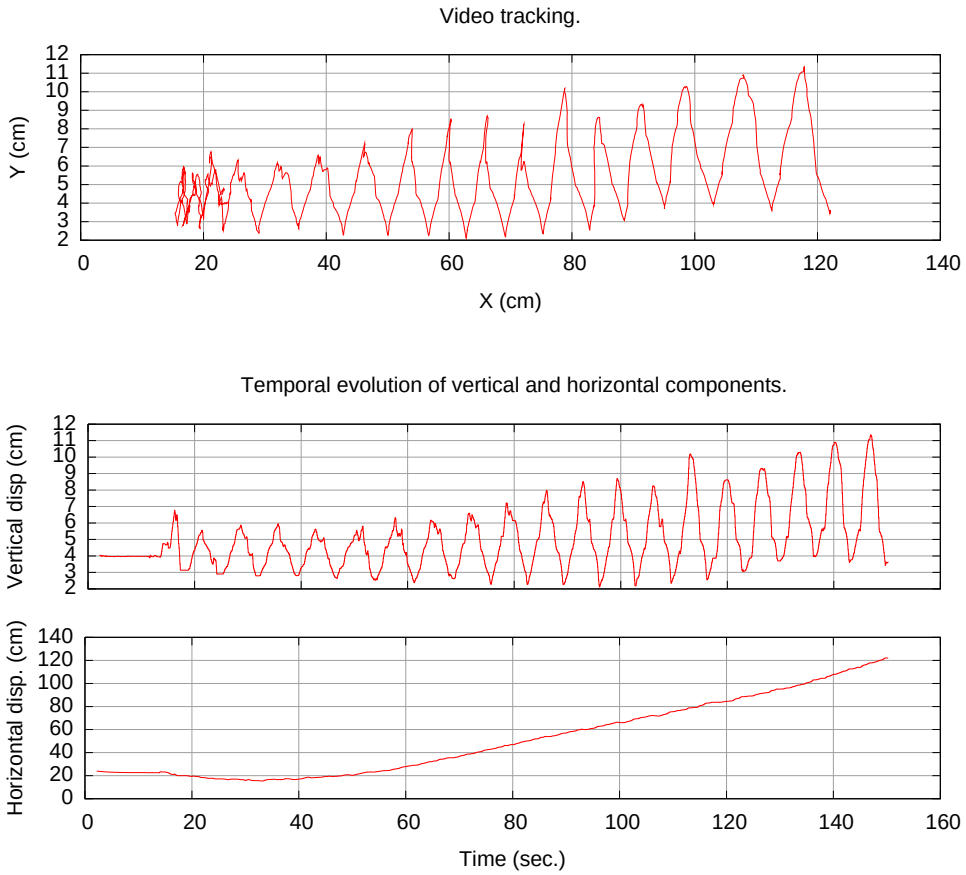


Figure 8.12: Asymmetrical inhibition initial transient behavior. The initial state of modules produces a backward locomotion in the robot. The topology handles this in a smooth way and slowly reverses the direction of travel to the appropriate one. A discontinuity in the platform of the simulation (see accompanying multimedia material) at about 80cm from the origin causes an artifact in the trajectory which results in irregular oscillations. Nevertheless, the result is a smooth forward locomotion with a steady speed, as the lower panel shows.

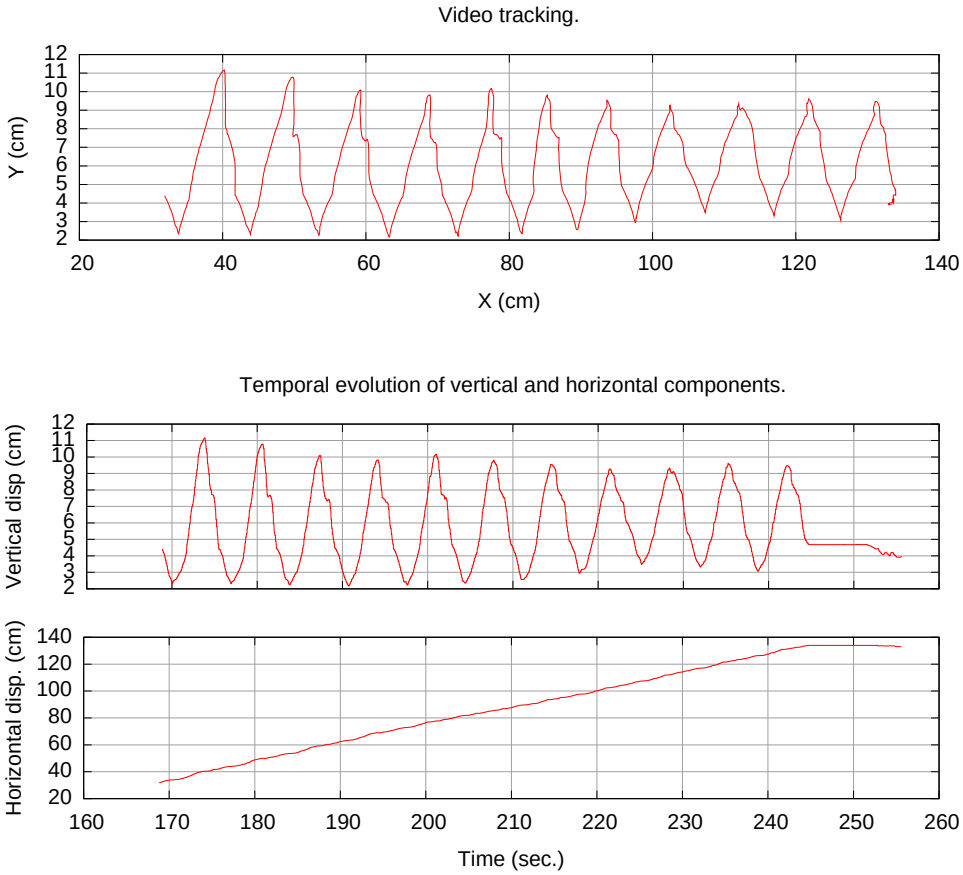


Figure 8.13: Asymmetrical inhibition stable locomotion regime, continued from figure 8.12. The robot proceeds with a stable forward locomotion. Again, at about 80cm from the origin locomotion is perturbed due to a discontinuity in the simulation platform. Nonetheless, forward speed is maintained.

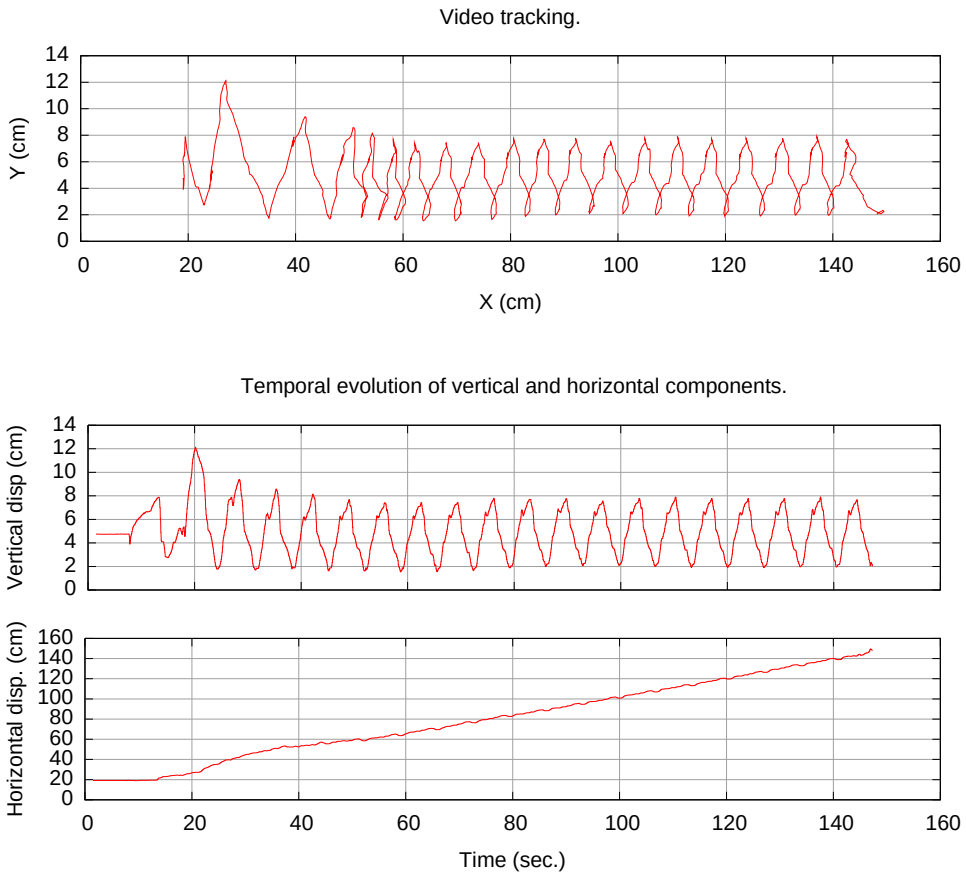


Figure 8.14: Inhibitory loop transient and stable locomotion. This connectivity pattern shows a very fast transient period. Neurons are initialized with arbitrary initial conditions, yet only a couple of oscillations after initialization the robot achieves a steady forward speed.

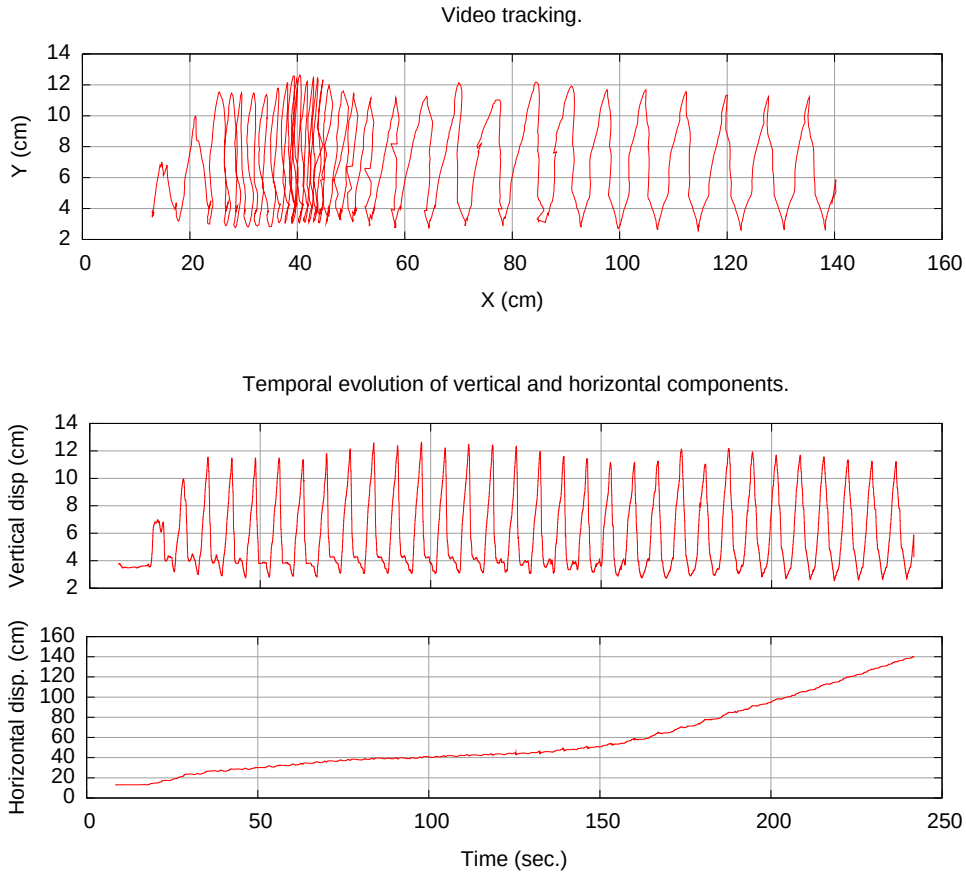


Figure 8.15: Bistable initial transient. This connectivity pattern shows a long transient period. The tracked module shows oscillations early from the beginning of the simulation, but a coordinated activity is still lacking. Until approximately 150s from the simulation there is no forward locomotion. The simulation was performed on a smooth surface on which the robot slips unless perfect coordination is achieved. This simulation would have performed better in the transient state on a rougher ground.

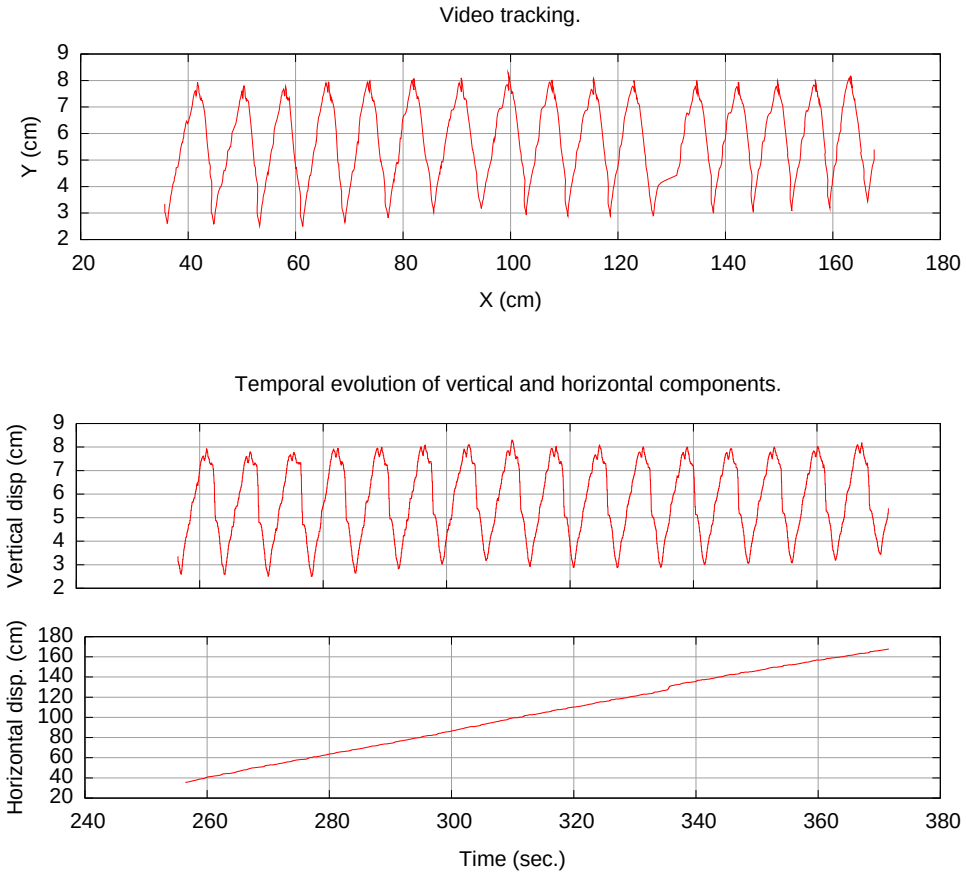


Figure 8.16: Bistable stable locomotion continued from figure 8.15. Once a global coordination has been established, the robot proceeds at a steady pace even on slippery surface on which transient activity had some problems.

In a different example, using the same elements and topology, we show expose the CPG to a very strong noise that completely disrupts its activity. After the perturbation is lifted, the CPG recovers its steady state oscillation, similarly to the previous case.

These are only two representative examples. From different simulations, we expect similar results using any of the different connectivity patterns from the previous chapter. Simulations have shown their ability to negotiate and maintain stable phase differences among modules. Besides, they are formulated on general, scalable grounds, and implemented following the same dynamical principles. The proposed mechanisms can also be successfully used to build CPG controllers for rhythmic activity in non-modular robots.

Part III

Summary and conclusion (english and spanish)

9 Summary of results

- We have built a modular oscillator with bursting neurons and inhibitory synapses following the half-center concept. It is an example of the minimal non-open topology that implements winner-less competition dynamics. In isolation, each neuron produces a stable and robust bursting sequence with a precise period. However, when coupled to other neurons with the right combination of synaptic dynamics and topology, they adjust their rhythm to produce a coordinated activity (section 6.3.1).

- Bursting in the neuron model employed in this thesis is the result of the combination of two dynamical systems with two different time scales. One is a slow oscillatory system. The other one is a fast oscillatory system that undergoes a bifurcation depending on the state of the slow subsystem. This combination can be exploited to multiplex a synchronization mechanism and an information carrying mechanism in the membrane potential signal of neurons. In fact, we use the slow subsystem as the basic mechanism for synchronization and the fast subsystem as the mechanism to encode locomotion information. The frequency of the slow subsystem encodes the frequency of oscillation, while the frequency and duration of the oscillations in the fast subsystem encode angular velocity and duration of movement (section 6.1).

- Combining bursting intrinsic neural dynamics and threshold synapses we implement a robust communication mechanism. We argue that both our model as well as living CPGs take advantage of bursting dynamics and threshold synapses to implement an event-based communication mechanism. This would help reducing the dependence of communication on precise values of a given biophysical magnitude, and instead relies on a higher level encoding mechanism (section 6.3.3).

- The oscillator can work in a wide range of frequencies depending on intrinsic neural and synaptic dynamics. The overall period of the oscillator depends linearly on the parameter controlling bursting period

in the neuron model. The most reliable and flexible configuration is for neurons to incorporate synaptic dynamics into their fast and slow subsystems simultaneously (section 6.3.1).

- Entrainment between a servo and our oscillator is possible using servo position error as neural feedback. A synaptic current proportional to that error is injected into one neuron, and an inversely proportional current is injected into the other neuron of the oscillator. The module produces a sustained oscillation whose frequency depends on the responsiveness of the servo. We show that the oscillator adapts its rhythm to the servo over several orders of magnitude without changing any of its parameters (section 6.3.2).

- We propose four different first-neighbor connectivity patterns that can be used to build modular CPG controllers. We focus on the implementation of dynamical invariants such that the proposed patterns are formulated independently of the actual parameters of implementation. All of the proposed topologies provide a mechanism for modules to negotiate the overall rhythm with their neighbors, each topology being based on a different principle. Their transient dynamics ensure that any perturbation will be effectively tackled and the system will eventually reach an autonomous, decentralized agreement to return to its nominal locomotion regime (section 7).

- We propose a flexible connectivity pattern based on mutual inhibition between first neighbors. The traveling direction is shown to be governed by the difference between the synaptic conductances in the upper pathway and the lower pathway (section 7.4).

- We propose a phase locking mechanism based on an inhibitory loop. Neighbor modules interconnected with this mechanism show fast transient dynamics leading to a stable phase difference (section 7.5).

- We propose a phase locking mechanism based on the simultaneous activation of excitatory and inhibitory synapses. The current injected into a neuron from an activated synapse depends on the activation level of the synapse, its conductance and the difference between the reversal potential of the synapse and the actual membrane potential of the neuron. This last difference introduces a dependence of synaptic current on neuron phase. The simultaneous activation of both synapses drives the interconnected neurons to a phase-locked oscillation (section 7.6).

- We propose a sequence enforcing mechanism based on a bistable neuron model and strong inhibition. Bistability is used as a temporal memory where each module can store the most recent state of its neighbors. Bistable neurons then serve as a flag for each module to indicate whether activation is or not appropriate in order to keep an effective sequence. We show that this pattern produces flexible transient dynamics leading to stable steady state synchronization (section 7.7.2).

- We analyze the integration of a CPG and a real worm-like modular robot. From random initial conditions, the intrinsic dynamics of neurons and the topology of the circuit work together in a negotiating effort to establish a global, decentralized coordination. After a strong perturbation of the rhythm, the CPG is able to restore locomotion in the robot. The locomotion encoding scheme and the motoneurons effectively generate a signal that can drive the modular robot (section 8).

10 Conclusion

Bio-inspiration is the process by which engineers build systems that incorporate ideas observed in natural living systems. One of the reasons in so doing is that nature offers solutions to problems shared by *natural* and *artificial* beings. Also, the individual human mind is limited in its possibilities to create something new. The events experienced by one single human individual in its lifespan cannot compare to the vast amount of experiments that nature has performed throughout the history of evolution.

Laying down bridges to transfer knowledge from one field to another requires a process of abstraction, where the essential aspects of a system are discriminated and abstracted from domain specific details.

A seminal work in the definition of a structured language to express problems and their solutions in an abstract way is (Alexander et al., 1977). Being an architect, Alexander realized that the process of designing buildings means providing common solutions to recurring problems. For instance, all buildings need to provide access to persons into the building, they have to provide enough sources of natural light and so on. But in practice, each individual construction must be implemented in a specific, unique context. So the great insight of Alexander was to create a language in which to express the common essence of different *instances* of a problem in a way independent of the details of its realization. He called that a *pattern language*.

The concept extended also to software engineering (Gamma et al., 1994). This work is a collection of solutions to recurring design problems, but they are expressed in a language independent of the actual details of the application. No specific programming language is required and no particular machine is specified to solve the described problems.

Bio-inspiration still has the need of a pattern language to describe elements and strategies in an abstract way. Recent theoretical and experimental evidence on living CPGs have proved the existence of general mechanisms that are key to an autonomous coordinated activity Rabinovich et al. (2006); Selverston et al. (2000). Rich intrinsic dynamics of building blocks, non-open topologies, mutual inhibition, these are all abstract elements and strategies, independent of the actual implementation that can contribute to the defini-

tion of such a language.

In this thesis we have made an emphasis on a qualitative approach, searching for general, qualitative mechanisms to build autonomous modular CPGs. We have built an oscillator following a general principle called half-center oscillator. This is a general design scheme on which other authors have also based their oscillators (Matsuoka, 1987), with different implementation details, however.

We have implemented a locomotion encoding mechanism based on bursting dynamics. The details of how bursting dynamics is actually implemented are not restricted. For instance, nature implements a similar mechanism in its own biological language (Thuma et al., 2003), while we have implemented it using the language of dynamical systems.

The particular implementation of the oscillator made in this thesis shows several interesting properties, that we believe other implementations with similar dynamics will also share. First, the richness provided by the combination of different time scales makes the oscillator work in a wide range of its parameter space. Second, classical non-linear oscillators without dynamical bifurcations are limited in their ability to produce different rhythms. We show that entrainment between the joint and our module controller is possible over a wide range of oscillation frequencies, up to several orders of magnitude. The key mechanism is the ability of the bursting neurons to delay the fast system's bifurcation that terminates the burst. This results in a highly autonomous and flexible adaptability of the oscillator. Third, we show that bursting dynamics is a robust mechanism to encode locomotion in a noise tolerant way. The fact that the oscillator selects spike threshold crossing events as its information units makes it more independent of the exact shape of neurons, which results in greater tolerance to noise.

Finally, we have proposed different connectivity patterns to implement dynamical invariant preserving CPGs. CPGs built with these patterns preserve the sequence of activation of their modules. We have formulated such patterns independent of the details of their implementation, establishing the roots of the aforementioned pattern language. All the different elements proposed in this thesis can be used to build CPGs for rhythmic control in robots other than modular worm-like ones.

In Mark Yim's web page[1] we found a very insightful comment:

> In one invited talk on this work I was asked "What is the research contribution?" I responded that the taxonomy of locomotion was the contribution. The questionner then asked "Is that all?"

[1] `http://ai.stanford.edu/users/mark/generalization.html`

> I cannot stress enough the importance of a classification system. In any new branch of science, the first thing that must be done is the classification of the instances of subjects being studied. This is the foundation upon which the whole area of study is built.

Classification is a seemingly simple and irrelevant task. However, this process confronts the scientist face to face with the systems under study. For if a system is to be classified, that is, assigned a label that says "this system has this and that properties", that system must first be fully understood. The starting point of this thesis was the control of a homogeneous modular robot. From this simple statement, many conclusions can be drawn.

Modularity is the essence of reality. As human technology advances, we find that any previous system thought to be atomic, i.e. without further subdivisions, is, in fact, modular. The question is, then, if there exists a limit where a really fundamental and atomic *thing* exists, or whether we will find out that everything that exists is the result of one single substance interacting with itself.

There is no need, however, to delve into such deep philosophical questions. If we climb up the ladder of hierarchical organization in nature, we find more and more complex systems which are the result of interacting modules. There is an important question worth asking: what is the general rule, homogeneity or heterogeneity?

In my opinion, there is a complex balance between both. The sea, land and the atmosphere can be considered homogeneous. Even though all of them contain diverse elements with complicated dynamics, it is difficult to establish borders between their parts. Living beings, however, are organized into clearly distinguishable parts. The simplest of bacteria have different subsystems that interact to execute the different processes necessary to sustain life. My conclusion is that complexity fosters specialization, and therefore heterogeneity arises.

However, there are some organizational levels were homogeneity appears again. In complex living animals like mammals or birds, similar cells assemble into organs. For instance, liver cells are quite similar one to another, and so are cortical columns in the brain. Climbing up again the organizational ladder we find societies of homogeneous individuals that assume specialized roles to sustain their community.

Homogeneity and heterogeneity must be considered when designing controllers for modular robots. In the connectivity patterns presented in chapter 7, we have proposed mostly homogeneous topologies. We know that non-open topologies are necessary to build effective CPGs (Huerta et al.,

2001). Borders clearly pose a restriction on a regular connectivity pattern because it cannot be repeated beyond borders, leaving some modules open at the ends. This problem must be solved by introducing an irregularity, a symmetry breaking element. Therefore non-open topologies for modular robots need to be heterogeneous at some point.

Looking back once again at the natural world, we find another complex equilibrium between change and stability. Many, if not all systems that surround us strive to maintain their inertia. Basic physical laws mandate that the total of energy and matter always be preserved, and that the momentum of bodies be kept. Surprisingly also the relationships between systems seem to be subject to these basic laws of preservation. There is, however, a subtle current of change that challenges a perfectly silent universe, and living systems are the most perfect combination of both, using change as a mechanism to preserve life.

Living systems have evolved to accept a changing environment. They also change, and they perform actions that induce further changes, both on the outside as well as the inside, to preserve their lives. The ability to plan and perform actions that preserve life is what we call autonomy.

From all the different survival strategies found in nature, maybe those developed by animals are the most complex and sophisticated. Interestingly, nature has found a wonderful tool to acquire information from the surroundings and from the animal itself, to process it and to create actions. This tool is the nervous system.

In this thesis we have explored different strategies that contribute to the autonomy of living CPGs, and that are also useful when building autonomous robots. As stated before, a striking feature of life is the use of change, of dynamical systems, to ensure stability. It is not a surprise that the fundamental building blocks of the nervous system in animals are highly dynamic systems. But if neurons would only change in a regular and predictable way, there would be no room for diversity. The world is too complicated, and demands high skills from animals if they want to survive.

Autonomy must go hand in hand with adaptation. That is, actions must be performed taking into account the context and the goal to be achieved. So it is crucial for living systems to know how well their actions are being performed, if circumstances have changed or if their goal is being met. This is also true for artificial systems, so, once again, a bridge needs to be established.

In my personal opinion, there is a good reason why the brain is not a *gray soup*. The strategies found in the nervous system of animals contribute to creating highly successfuly autonomous and adaptable beings. The building blocks of the nervous system are dynamically rich, in the sense that they can choose from a wide repertoire of possible behaviours, that result mainly

from a sophisticated non-linear interaction of elements. Not only are their dynamics robust, but they are also flexible to change according to external inputs. This, together with an adequate information transportation network is one of nature's most powerful computation devices.

To conclude, advances in CPG research in recent years provide new bio-inspiration for robotic design. We believe that the proposed design elements can lead to CPG control paradigms for autonomous locomotion that are less architecture specific, and provide solutions that present wider working regions in the parameter space of the models. We plan to further investigate and take advantage of how biological systems are able to maintain their invariants, and incorporate these new ideas into autonomous robotic control.

The work presented in this thesis has many seeds to develop different branches for future work.

One of the main results of this thesis is showing the adaptability that results from neuron models with rich intrinsic dynamics. Using feedback from the controlled joint, a modular controller will adjust its frequency to the responsiveness of the servo. We believe that this ability will be maintained in the case of a full-length robot, and that we will be able to design general CPGs that will autonomously adapt to the working conditions of different robots. Besides, one of the benefits of modular robotics is the easy replacement of damaged parts. It might be the case then, that different modules are in different working conditions. We believe that the mechanisms proposed will be able to handle these situations and still achieve a global coordinated locomotion.

CPGs are complex information processing devices. The computations they are capable of performing depend on the richness and the complexity of their dynamics. We want to explore further dynamical mechanisms in order to generate more complex motor programs.

11 Resumen de resultados

- Hemos construido un oscilador modular con neuronas en ráfagas y sinapsis inhibitorias siguiendo el concepto de *half-center*. Se trata de un ejemplo de la topología mínima no abierta que implementa dinámica de competencia sin ganador. De forma aislada, cada neurona produce una secuencia estable y sólida con un período preciso. Sin embargo, cuando se acoplan a otras neuronas con la combinación correcta de dinámica sinápticas y topología, su ritmo se modifica para producir una actividad coordinada (sección 6.3.1).

- El comportamiento en ráfagas en el modelo de neurona empleado en esta tesis es el resultado de la combinación de dos sistemas dinámicos con dos escalas de tiempo diferentes. Uno es un sistema oscilatorio lento. El otro es un sistema oscilatorio rápido que se somete a una bifurcación dependiendo del estado del subsistema lento. Esta combinación puede ser explotada para multiplexar un mecanismo de sincronización y un mecanismo de transmisión de información en la señal de potencial de membrana de las neuronas en ráfagas. De hecho, el subsistema lento se utiliza como mecanismo básico de sincronización y el subsistema rápido como mecanismo para codificar la información de locomoción. La frecuencia del subsistema lento codifica la frecuencia de oscilación, mientras que la frecuencia y la duración de las oscilaciones en el subsistema rápido codifican la velocidad angular y la duración del movimiento (sección 6.1).

- Combinando dinámicas neuronales intrínsecas en ráfagas y sinapsis con umbral implementamos un mecanismo robusto de comunicación. Argumentamos que tanto nuestro modelo como los CPGs vivos utilizan la dinámica en ráfagas y las sinapsis con umbral para implementar un mecanismo de comunicación basado en eventos. Esto ayudaría a que la comunicación sea menos dependiente de los valores precisos de una magnitud biofísica dada, y que lo sea, a cambio, de un mecanismo de codificación de nivel superior (sección 6.3.3).

- El oscilador puede trabajar en una amplia gama de frecuencias en función de la dinámica neuronal intrínseca en ráfagas y de la dinámica sináptica. El período global del oscilador depende linealmente del parámetro que controla el período de la ráfaga en el modelo de neurona. La configuración más confiable y flexible para las neuronas en ráfagas es la que incorpora la dinámica sináptica en los subsistemas rápido y lento al mismo tiempo (sección 6.3.1).

- La adaptación entre un servo y nuestro oscilador es posible usando el error de posición del servo como retroalimentación neuronal. Una corriente sináptica proporcional a dicho error se inyecta en una neurona, y otra inversamente proporcional se inyecta en la otra neurona del oscilador. El módulo produce una oscilación cuya frecuencia depende de la capacidad de respuesta del servo. Se demuestra que el oscilador adapta su ritmo al del servo en varios órdenes de magnitud sin cambiar ninguno de sus parámetros (sección 6.3.2).

- Proponemos cuatro patrones de conectividad diferentes a primeros vecinos que pueden ser utilizados para construir controladores modulares CPG. Nos hemos centrado en la implementación de invariantes dinámicos de tal manera que los patrones propuestos estén formulados independientemente de los detalles reales de implementación. Todas las topologías propuestas constituyen un mecanismo mediante el cual los módulos pueden negociar un ritmo global con sus vecinos, y cada una de ellas está basada en un principio diferente. Sus dinámicas transitorias garantizan que cualquier perturbación será absorbida de forma efectiva y que el sistema llegará finalmente a un acuerdo autónomo y descentralizado para regresar a su régimen nominal de locomoción (sección 7).

- Proponemos un patrón de conectividad flexible basado en inhibición mutua entre primeros vecinos. Demostramos que la dirección de desplazamiento se rige por la diferencia entre las conductancias sinápticas en la vía superior y la vía inferior (sección 7.4).

- Proponemos un mecanismo de bloqueo de fase basado en un bucle inhibidor. Módulos vecinos interconectados con este mecanismo muestran dinámicas transitorias rápidas que llevan a una diferencia de fase estable (sección 7.5).

- Proponemos un mecanismo de bloqueo de fase basado en la activación simultánea de sinapsis excitatorias e inhibitorias. La corriente que una

sinapsis inyecta en una neurona depende del nivel de activación de la sinapsis, de su conductancia y de la diferencia entre su potencial de inversión y el potencial de membrana de la neurona. Esta última diferencia introduce una dependencia entre la corriente sináptica y la fase de la neurona postsináptica. La activación simultánea de ambas sinapsis lleva a las neuronas interconectadas a una oscilación bloqueada en fase (sección 7.6).

- Proponemos un mecanismo de imposición de secuencias basado en un modelo de neurona biestable e inhibición fuerte. La biestabilidad se utiliza como un mecanismo de memoria temporal, donde cada módulo puede almacenar el estado más reciente de sus vecinos. Las neuronas biestables pueden servir como un indicador para que cada módulo señale si la activación de la promotora o la remotora es apropiada o no. Demostramos que este patrón de conectividad produce dinámicas transitorias flexibles que conducen a un estado de sincronización estable (sección 7.7.2).

- Analizamos la integración de un CPG con un robot gusano modular real. Partiendo de condiciones iniciales arbitrarias, la dinámica intrínseca de las neuronas en ráfagas y la topología del circuito trabajan juntas para negociar y establecer una coordinación global y descentralizada. Después de una perturbación fuerte del ritmo, el CPG es capaz de restaurar la locomoción en del robot. El esquema de codificación de la locomoción y las motoneuronas efectivamente generan una señal que puede conducir al robot modular (sección 8).

12 Conclusión

La bioinspiración es el proceso mediante el cual los ingenieros construyen sistemas que incorporan ideas observadas en los sistemas vivos naturales. Una de las razones para ello es que la naturaleza ofrece soluciones a problemas comunes entre los seres *naturales* y *artificiales*. Además, la mente humana individual está limitada en sus posibilidades para crear algo nuevo. Los sucesos experimentados por un único individuo humano en su vida no pueden ser comparados con la gran cantidad de experimentos que la naturaleza ha llevado a cabo a lo largo de la historia de la evolución.

Establecer puentes para transferir conocimiento de un campo a otro requiere un proceso de abstracción, donde los aspectos esenciales de un sistema sean discriminados y abstraídos de los detalles específicos del dominio.

Un trabajo fundamental en la definición de un lenguaje estructurado para expresar problemas y sus soluciones de una manera abstracta es (Alexander et al., 1977). Siendo un arquitecto, Alexander se dio cuenta de que el proceso de diseño de edificios implica proporcionar soluciones comunes a problemas recurrentes. Por ejemplo, todos los edificios necesitan proporcionar acceso a las personas, proporcionar suficientes fuentes de luz natural y demás. Pero en la práctica, cada obra individual debe ser implementada en un contexto único y específico. Así que la gran visión de Alexander fue crear un lenguaje con el que expresar la esencia común de diferentes *instancias* de un mismo problema, de una manera independiente de los detalles de su realización. A esto lo llamó un *lenguaje de patrones*.

El concepto se extendió también a la ingeniería del software (Gamma et al., 1994). Este trabajo es una colección de soluciones a problemas recurrentes de diseño, expresados en un lenguaje independiente de los detalles reales de la aplicación. No se requiere ningún lenguaje de programación específico ni ninguna máquina en particular para resolver los problemas descritos.

La bioinspiración tiene todavía la necesidad de un lenguaje de patrones para describir elementos y estrategias de una manera abstracta. Evidencias recientes tanto teóricas como experimentales sobre CPGs vivos han demostrado la existencia de mecanismos generales fundamentales para una ac-

tividad autónoma y coordinada Rabinovich et al. (2006); Selverston et al. (2000). Una dinámica intrínseca rica de los elementos del circuito, topologías no abiertas, inhibición mutua, todos estos ingredientes son estrategias y elementos abstractos, independientes de la aplicación final, que pueden contribuir a la definición de este lenguaje.

En esta tesis se ha hecho hincapié en un enfoque cualitativo, buscando mecanismos generales y cualitativos para construir CPGs modulares autónomos. Hemos construido un oscilador siguiendo un principio general llamado oscilador *half-center*. Se trata de un esquema de diseño general en el que otros autores también han basado sus osciladores (Matsuoka, 1987), con diferentes detalles de implementación, sin embargo.

En este trabajo hemos implementado un mecanismo de codificación de la locomoción basado en una dinámica neuronal en ráfagas. Los detalles de cómo se implementa realmente esta dinámica no están limitados. Por ejemplo, la naturaleza implementa un mecanismo similar en su propio lenguaje biológico (Thuma et al., 2003), mientras que nosotros lo hemos implementado utilizando el lenguaje de los sistemas dinámicos.

La implementación particular del oscilador realizado en esta tesis muestra varias propiedades interesantes, que creemos que otras implementaciones con una dinámica similar también compartirán. En primer lugar, la riqueza proporcionada por la combinación de diferentes escalas de tiempo hace que el oscilador pueda trabajar en una amplia gama de su espacio de parámetros. En segundo lugar, los osciladores clásicos no lineales sin bifurcaciones están limitados en su capacidad para producir ritmos diferentes. Demostramos que el acoplamiento entre la articulación y nuestro controlador modular es posible en un rango de frecuencias de oscilación amplio, hasta de varios órdenes de magnitud. El mecanismo clave es la capacidad de las neuronas en ráfagas para retrasar la bifurcación del subsistema rápido que termina la ráfaga. Esto se traduce en una gran capacidad de adaptación autónoma y flexible del oscilador. En tercer lugar, mostramos que esta dinámica en ráfagas es un mecanismo robusto para codificar la locomoción de una manera tolerante al ruido.

Por último, hemos propuesto diferentes modelos de conectividad para implementar CPGs que preserven invariantes dinámicos. Los CPGs construidos con estos patrones mantienen la secuencia de activación de sus módulos. Hemos formulado tales patrones independientemente de los detalles de su implementación, estableciendo las raíces del mencionado lenguaje de patrones. Todos los elementos propuestos en esta tesis se pueden utilizar para construir CPGs para el control rítmico en otros robots modulares además de los que tienen forma de gusano.

En la página web de Mark Yim [1] encontramos un comentario muy iluminador (traducido del inglés):

> En una charla a la que me invitaron para hablar de este trabajo me preguntaron "¿cuál es su contribución a la ciencia?" Respondí que la contribución era la taxonomía de locomoción . El otro me preguntó de nuevo: "¿eso es todo?"
>
> No puedo enfatizar suficientemente la importancia de un sistema de clasificación. En cualquier rama nueva de la ciencia, lo primero que debe hacerse es la clasificación de los objetos de estudio. Ésta es la base sobre la que se sustentará todo el área de conocimiento.

Clasificar es una tarea aparentemente simple e irrelevante. Sin embargo, este proceso confronta al científico con los sistemas que ha de estudiar. Porque para clasificar un sistema, es decir, para asignarle una etiqueta que diga "este sistema tiene esta y aquella propiedades", ese sistema debe ser plenamente comprendido primero. El punto de partida de esta tesis ha sido el control de un robot modular homogéneo. De esta simple declaración, se pueden sacar muchas conclusiones.

La modularidad es la esencia de la realidad. A medida que avanza la tecnología humana, encontramos que cualquier sistema que se creyera atómico, es decir, sin subdivisiones, es, de hecho, modular. La pregunta es, entonces, si existe un límite donde existe una *cosa* realmente fundamental y atómica, o si por el contrario finalmente descubriremos que todo lo que existe es el resultado de la interacción de una única sustancia consigo misma.

Sin embargo no es necesario entrar en cuestiones filosóficas tan profundas. Si subimos por los distintos niveles de organización jerárquica de la naturaleza, nos encontramos con sistemas cada vez más complejos, que son el resultado de la interacción de sus módulos. Hay una pregunta importante que vale la pena hacerse: ¿cuál es la regla general, la homogeneidad o la heterogeneidad?

En mi opinión hay un complejo equilibrio entre ambas. El mar, la tierra y la atmósfera se pueden considerar homogéneos. A pesar de que todos ellos contienen diversos elementos con una dinámica complicada, es difícil establecer fronteras entre sus partes. Los seres vivos, sin embargo, se organizan en partes claramente distinguibles. La más simple de las bacterias tiene diferentes subsistemas que interactúan para ejecutar los diferentes procesos necesarios para mantener la vida. Mi conclusión es que la complejidad fomenta la especialización, y de ahí surge la heterogeneidad.

[1] `http://ai.stanford.edu/users/mark/generalization.html`

Sin embargo, en algunos niveles de organización aparece la homogeneidad otra vez. En animales vivos complejos como los mamíferos o las aves, las células similares se unen para formar órganos. Por ejemplo, las células del hígado son bastante similares entre sí, y también lo son las columnas corticales en el cerebro. Subiendo otra vez en la escalera organizativa nos encontramos con sociedades de personas homogéneas que asumen funciones especializadas para sostener su comunidad.

La homogeneidad y la heterogeneidad deben tenerse en cuenta al diseñar controladores para robots modulares. En los patrones de conectividad presentados en el capítulo 7, hemos propuesto sobre todo topologías homogéneas. Sabemos que las topologías no abiertas son necesarias para construir CPGs efectivos (Huerta et al., 2001). Los bordes representan claramente una restricción para un patrón de conectividad regular porque no se puede repetir el patrón más allá de ellos, dejando algunos módulos abiertos en los extremos. Este problema debe resolverse mediante la introducción de alguna irregularidad, algún elemento que rompa la simetría. Por lo tanto las topologías no abiertas para robots modulares deben ser heterogéneas en algún punto.

Mirando una vez más al mundo natural, nos encontramos con otro complejo equilibrio entre cambio y estabilidad. Muchos, si no todos los sistemas que nos rodean se esfuerzan por mantener su inercia. Las leyes físicas básicas dictan que el total de energía y la materia se ha de conservar siempre, y que el momento de los cuerpos se ha de mantener. Sorprendentemente también las relaciones entre los sistemas parecen estar sujetas a estas leyes básicas de conservación. Hay, sin embargo, una corriente sutil de cambio que es un desafío a un universo perfectamente silencioso, y los sistemas vivos son la combinación más perfecta de ambos, utilizando el cambio como un mecanismo de preservación de la vida.

Los sistemas vivos han evolucionado para aceptar un entorno cambiante. Ellos también cambian, y llevan a cabo acciones que inducen nuevos cambios, tanto en el exterior como en el interior, para preservar sus vidas. La capacidad de planificar y llevar a cabo acciones para preservar la vida es lo que llamamos autonomía.

De todas las diferentes estrategias de supervivencia que se encuentran en la naturaleza, tal vez sean las desarrolladas por los animales las más complejas y sofisticadas. Curiosamente, la naturaleza ha encontrado una herramienta maravillosa para adquirir información del entorno y del propio animal, procesarla y crear acciones. Esta herramienta es el sistema nervioso.

En esta tesis hemos explorado diferentes estrategias que contribuyen a la autonomía de los CPGs vivos, y que también son útiles para la construcción de robots autónomos. Como se dijo antes, una característica sorprendente de la vida es el el uso del cambio, de los sistemas dinámicos, para garantizar

la estabilidad. No es una sorpresa que los componentes fundamentales del sistema nervioso en los animales sean sistemas altamente dinámicos. Pero si las neuronas sólo cambiasen de una manera regular y predecible, no habría espacio para la diversidad. El mundo es demasiado complicado, y requiere de grandes habilidades por parte de los animales para sobrevivir.

La autonomía debe ir de la mano con la adaptación. Es decir, las acciones deben llevarse a cabo teniendo en cuenta el contexto y la meta que se quiere alcanzar. Por lo tanto, es crucial que los sistemas vivos conozcan la efectividad con la que sus acciones se están llevando a cabo, si las circunstancias han cambiado o si su objetivo se está cumpliendo. Esto también es válido para sistemas artificiales, por lo que, de nuevo, es necesario establecer un puente.

En mi opinión, hay una buena razón por la que el cerebro no es una *sopa gris*. Las estrategias que se encuentran en el sistema nervioso de los animales contribuyen a crear seres altamente autónomos y adaptables. Los elementos fundamentales del sistema nervioso son dinámicamente ricos, en el sentido de que pueden elegir entre un amplio repertorio de comportamientos posibles, que son el resultado principalmente de una sofisticada interacción no lineal de elementos. No sólo tienen una dinámica robusta, sino que también son flexibles para cambiar de acuerdo a entradas externas. Esto, junto con una red de transporte de información adecuada resulta en uno de los dispositivos de cómputo más poderosos de la naturaleza.

Para concluir, los avances en la investigación de los CPGs en los últimos años ofrecen una nueva bioinspiración para el diseño robótico. Creemos que los elementos de diseño propuestos pueden llevarnos hacia paradigmas de control CPG para locomoción autónoma menos dependientes de la arquitectura específica, y que ofrecen soluciones que presentan mayores regiones de trabajo en el espacio de parámetros de los modelos . Nuestro plan es investigar más a fondo y aprovecharnos de la forma en que los sistemas biológicos son capaces de mantener sus invariantes, e incorporar estas nuevas ideas al control de robots autónomos.

El trabajo presentado en esta tesis tiene muchas semillas para desarrollar diferentes ramas en un trabajo futuro. Uno de los principales resultados de esta tesis es demostrar la capacidad de adaptación que resulta de tener neuronas con una dinámica intrínseca rica. Usando información sobre la articulación controlada, un controlador modular ajusta su frecuencia en función de la capacidad de respuesta del servo. Creemos que esta capacidad se mantendrá en el caso de un robot completo, y que seremos capaces de diseñar CPGs generales que de manera autónoma se adapten a las condiciones de trabajo de diferentes robots. Además, uno de los beneficios de la robótica modular es que la sustitución de partes dañadas es muy sencilla. Podría darse el caso, entonces, de que diferentes módulos estén en diferentes

condiciones de trabajo. Creemos que los mecanismos propuestos serán capaces de manejar estas situaciones y aún así lograr una locomoción global coordinada.

Part IV

Appendix

13 A dynamical systems simulation library

We have developed a C++ library that eases the creation of dynamical neural networks, with which the Central Pattern Generators (CPGs) for this thesis have been implemented. Special attention has been paid to the ease with which new models can be added to the library, and with which networks of elements described by this models can be specified.

The library offers different components:

- Wrappers
- Models
- Concepts
- Integrators
- Synapses

In the following we describe the function of each module.

13.1 Model definition

The library's central goal is to simulate networks of interconnected dynamical systems. Therefore, specifying new dynamical models should pose as little strain on the final user as possible.

A dynamical model is usually formulated as a set of differential equations that govern the change rate of some variables, possibly influenced by some parameters:

$$\dot{\vec{x}} = F(\vec{x}, \vec{p}) \tag{13.1}$$

with $\vec{x} = \{x_1, \ldots, x_n\}$ and $\vec{p} = \{p_1, \ldots, p_m\}$.

The user should write as little code as possible in order to write a new model that can be used together with the library. Then, the library will

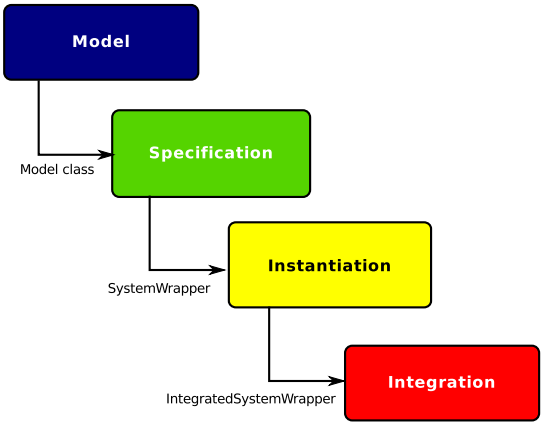


Figure 13.1: Separation of concerns within the library: the user provides a class containing the necessary elements to describe the model; using the `SystemWrapper` class, storage space is reserved and getter/setter methods are generated; finally, using the `IntegratedSystemWrapper` class, an integrator may be bound to the model that will provide a user-transparent `step` method.

provide generic components to aid in writing common code (see figure 13.1). At the very least, he/she should provide:

- A list of the variables of the model
- A list of the parameters of the model
- The equations of the model

Algorithm 13.1 shows an example of a model specification. This example includes all the necessary information for the library to be able to integrate an harmonic oscillator:

Variables: an enumeration defines the type MyModel::variable which can take on three values: x and y, or n_variables. A useful fact is that C++ enums may be converted to integers, but not vice-versa. So, expressions of the type Neuron::x may be used to index a particular array to access the values of the model's variable. Furthermore, the fact that each enum represents an independent type allows the use of polymorphism and prevents the final user from some semantic errors as trying to access a variable that a particular model does not implement[1].

[1]Note that in this context, ModelA::x and ModelB::x are two distinct model variables, even though their names are equal. That is, scopes apply.

Listing 13.1: A sample model

```
class MyModel
{
public:
    enum variable{x, y, n_variables};
    enum parameter{w, n_variables};

    void eval(const Precission * const vars,
              Precission * const params,
              Precission * const incs) const
    {
        incs[x] = params[w] * vars[y];
        incs[y] = -params[w] * vars[x];
    }
};
```

Parameters: in the same fashion as variables, a type MyModel::parameter is defined for parameters of the model.

Equations: function eval provides the necessary calculations for integrators to evaluate the right-hand-side of a dynamical model. Equations are expressed in a natural way, taking into account that variables and parameters must be referenced through the vars[v] and params[p] syntax, and that the result must be stored using the incs[v] syntax.

This class already provides the necessary elements to interface with the rest of the library, as we explain in later sections of the paper.

13.2 Mix-in based programming

Users of our library need only write very simple dynamical models specifications. Then, the library provides mechanisms to enhance model classes and equip them with sophisticated behavior such as memory allocation, step integration and serialization, for example.

A common design pattern employed when adding functionality to an already existing implementation is the *decorator* Gamma et al. (1993) pattern. A decorator class is typically a member of a family of classes. It can *decorate* any other member of the family, meaning that this class will forward most of the messages it receives to the decorated class untouched, but it will reinterpret, and possibly add some functionality to some specific messages. The restrictions are that the decorated class must derive from a specific type, and usually that the decorator class also derives from the same family base type. This way decorations become *stackable*.

Listing 13.2: Mix-in class structure

```
template <typename Base>
class Wrapper : public Base
{
protected:
    protected_enhancement;

public:
    public_enhancement;

    typedef typename Base::type_feature type_feature;
};
```

There are some restrictions that make this approach sub-optimal in the sense of flexibility:

- Classes that are to be decorated must explicitly be derived from a particular base class.
- Classes that are to be decorated must be instantiable, since decoration happens at the object level, not class level.
- Type information from the decorated object cannot be used to configure the decorating class.
- The system depends upon run-time polymorphism.

In order to provide the end user with the maximum flexibility without the loss of performance possibly incurred by using run-time polymorphism, a different approach was taken in our library. We provide a set of generic decorator classes that can be used to extend any class that implements the appropriate interface, irrespective of its inheritance tree. This kind of generic decorator is termed a *mix-in* (Smaragdakis and Batory, 2001). Listing 13.2 shows the structure of a basic mix-in class. The basic property of such a class is that it inherits from the class that is passed as template argument. The benefit of this is that extensions can be written independently of the extended classes, type information may be reused and is fully available at compile time, and the final user is still free to stack them as she sees fit.

Listing 13.3: Mix-in based programming

```
class SimpleClass
{
protected:
    protected_behaviour;

public:
    public_behaviour;
};

typedef Wrapper<SimpleClass> EnhancedClass;
```

13.3 Wrappers

13.3.1 SystemWrapper

Description This *wrapper* provides the capacity of instantiating a given model, allocating memory space to hold its state (variables) and parameters, and methods to access and modify their value.

Requisites The extended class must provide:

- The same requisites imposed for a model class as that shown in 13.1

Working The SystemWrapper class:

- Provides a nested class called ConstructorArgs used to instantiate the class and pass the corresponding parameters for the model.
- Provides a non-default constructor that takes in an argument of type ConstructorArgs and copies the parameter values contained therein.
- Creates an array of type precission_t and size n_parameters.
- Creates an array of type precission_t and size n_variables.
- Provides methods to access variables and parameters like the following:
- Will inherit all attributes and methods from the wrapped class, and will forward define the type precission_t

```
precission_t get(variable var) const
{
   return m_variables[var];
}
```

```
void set(variable var, precission_t value)
{
   m_variables[var] = value;
}

precission_t get(parameter param) const
{
   return m_parameters[param];
}

void set(parameter param, precission_t value)
{
   m_parameters[param] = value;
}
```

Advantage is taken of the fact that models already provide enumeration types describing their variables and parameters.

13.3.2 IntegratedSystemWrapper

Description This *wrapper* binds a system and an integrator, so that a single call to the step `step()` method will advance the system a given amount of time, performing the necessary calculations using the integrator.

Requisites The extended system class must provide:

- An interface similar to that provided by `SystemWrapper`.
- A method with signature `void pre_step(precission_t)`, that will be called before performing the integration.
- A method with signature `void post_step(precission_t)`, that will be called after performing the integration.

Working The `IntegratedSystemWrapper` class:

- Forward defines all nested types defined in the wrapped class.
- Preserves the constructor method defined by `SystemWrapper`.
- Provides a `step(precission_t h)` method that will integrate the system and advances it `h` units of time.

13.3.3 SerializableWrapper

Description This *wrapper* provides simple storage and retrieval capabilities to a system. Using this wrapper, a system may be saved to disk to be later reloaded with the same state and same parameter values.

Requisites The extended system class must provide:

- An interface similar to that provided by SystemWrapper.

Working The SerializableWrapper class:

- Forward defines all nested types defined in the wrapped class.
- Preserves the constructor method defined by SystemWrapper.
- Provides a save(std :: ostream &os) method that will serialize variables and parameters and will write them to the stream os.
- Provides a load(std :: istream &is) method that will read variables and parameters from the stream is, in the same order as the save() method.

13.3.4 TimeWrapper

Description This *wrapper* is useful when coupling systems with different time scales, and one should be integrated more often than others. For each *global* time step, the wrapper will integrate the system more than one step.

Requisites The extended system class must provide:

- An interface similar to that provided by IntegratedSystemWrapper.

Working The TimeWrapper class:

- Forward defines all nested types defined in the wrapped class.
- Provides a non-default constructor that takes in one argument of type ConstructorArgs and one optional argument of type **int**, with default value 1. The ConstructorArgs parameter will be passed to the constructor of the wrapped class.
- Provides a wrapper method **void** step(precission h) that will call the wrapped class's step() method as many times as the second argument to the constructor specifies.

13.4 Concepts

Wrappers expect some particular features from the extended class, that is, they need that the extended class conform to a given interface. There is, unfortunately, no native mechanism in C++ that allows for such conformance testing, so it must be done programmatically. We have used the technique proposed by Siek et al. (Siek and Lumsdaine, 2000), as implemented within the *boost*[2] library.

In short, according to this technique, the way to perform interface conformance testing is to implement a *Concept* class that uses the tested class, such as instantiating it, instantiating variables of defined types, etc. Then the compiler is tricked into compiling this class without actually generating any code for it. If the tested class does not have the features demanded by the *concept*, compilation will bail out with an error. As an example, we try to compile the following code using g++ 4.2:

```
#include <wrappers/SystemWrapper.h>

class Model
{
};

typedef SystemWrapper<Model> MySystem;

int main()
{
  MySystem ms;

  return 0;
}
```

the compiler will yield the following output:

```
/usr/local/include/boost/mpl/if.hpp:67:   instantiated from
'boost::mpl::if_<boost::concept::not_satisfied<ModelConcept<Model> >,
boost::concept::constraint<ModelConcept<Model> >,
boost::concept::requirement<ModelConcept<Model> > >'
/usr/local/include/boost/concept/detail/general.hpp:19:   instantiated
from 'boost::concept::requirement_<void (*)(ModelConcept<Model>)>'
/home/elferdo/local/NeuralNetworks/0.3/wrappers/SystemWrapper.h:57:
instantiated from 'SystemWrapper<Model>'
bad_model.cpp:11:   instantiated from here
```

[2] http://www.boost.org

```
.../NeuralNetworks/0.3/wrappers/../concepts/ModelConcept.h:48:
error: no type named 'precission_t' in 'class Model'

.../NeuralNetworks/0.3/wrappers/../concepts/ModelConcept.h:54:
error: 'n_variables' is not a member of 'Model'

.../NeuralNetworks/0.3/wrappers/../concepts/ModelConcept.h:55:
error: 'n_parameters' is not a member of 'Model'

.../NeuralNetworks/0.3/wrappers/../concepts/ModelConcept.h:56:
error: 'const class Model' has no member named 'eval'
```

The four unmet demands that SystemWrapper poses on the Model class are clearly identified, so the user can easily fix them.

13.5 Integrators

An integrator is a function that takes in as arguments a set of variables, a set of parameters and a function representing the equations of a dynamical system, and returns the value of the variables after a given time interval has elapsed.

We want to write generic integrators, that may be used to integrate arbitrary systems. Integrators may be used as standalone functions or as a class. The user may even integrate systems that are not implemented using the wrapper functionality of this library. We do not want to incur the performance penalty associated with run-time polymorphism, so we choose to use the compile-time type deduction facility that C++ provides, and use generic functions instead.

Integrator functions are specified as follows (this is a simplification of the actual implementation):

However, they are encapsulated in classes to ease their integration with the rest of the library, most notably through the IntegratedSystemWrapper wrapper:

The step method is the one that performs the integration of the system. It is declared **static**, so it may be called standalone, without any object. As a first argument, we need to pass a system object. Then, the amount of time to integrate the system, an array holding the state variables of the system and, finally, an array holding the parameters of the system.

If the function is to be invoked standalone, they System type must be explicitly specified:

From that argument its type is deduced, so that its features may be used inside the function. For instance, the System type must specify the precision of

Listing 13.4: Integrator function skeleton

```
template <typename System>
void step(System &s,
          typename System::precision_t h,
          typename System::precision_t *vars,
          typename System::precision_t *params)
{
  const int number_of_vars = typename System::n_variables;

  typename System::precision_t temp_vars[number_of_vars];

  /* Do calculations and update vars */
}
```

Listing 13.5: Integrator class skeleton

```
class Integrator
{
    template <typename System>
    void step(System &s, ...)
    {
        ...
    }
}
```

Listing 13.6: Standalone integrator example

```
void hodgkin_huxley_eval(...)
{
...
};

double variables[];
double parameters[];

RungeKutta4<HodgkinHuxley>::step(my_neuron, 0.001,
```

Listing 13.7: Base class with non-default constructor and derived class

```
struct Base
{
    Base(int x){};
};

struct Derived: public Base
{
};

void main()
{
    Derived derived(3);
}
```

its variables (**double**, **float**, etc.) as a nested type System::precision_t. The nice thing here is that the compiler will synthesize a function with the appropriate types once such a function is invoked.

13.6 Known bugs and future work

In this section we propose enhancements to either solve known bugs, or to make the library more generic and more usable.

13.6.1 The constructor problem

One of the problems faced using mix-ins is that a class that inherits from another one does not inherit non-default constructors defined in the base class. Let's try the following example:

This yields the following error:

```
constructor.cpp:12: error: no matching function for call to
  'Derived::Derived(int)'
constructor.cpp:7: note: candidates are: Derived::Derived()
constructor.cpp:7: note:                 Derived::Derived(const
  Derived&)
```

However, if we try to construct an object of type Derived using one of the available default constructors, proposed by the compiler:

The following error will ensue:

```
constructor.cpp:7: error: no matching function for call to
  'Base::Base()'
```

Listing 13.8: Base class with non-default constructor and derived class

```
void main()
{
    Derived derived;
}
```

Listing 13.9: Base class and derived class with explicit constructor

```
struct Base
{
    Base(int x){};
};

struct Derived: public Base
{
    Derived() : Base(3)
    {
    }
};

void main()
{
    Derived derived;
}
```

```
constructor.cpp:3: note: candidates are: Base::Base(int)
constructor.cpp:2: note:                 Base::Base(const Base&)
```

If a class provides a non-default constructor and does not provide a default one, then the compiler will neither provide it. So trying to *default construct* Derived will also try to default construct Base. But this will fail, because there is no default constructor for that class.

The only solution to this problem is for Derived to provide a constructor that explicitly passes the required arguments to Base's constructor:

This will compile without any problem.

In the case of wrappers, the problem lies in how a wrapper class guesses the signatures of the constructors of the wrapped class. The case is illustrated as follows:

The solution that we propose is similar to one of the solutions proposed in (Eisenecker et al., 2000). The SystemWrapper provides a ConstructorArgs class that gets propagated through the wrapping chain. This class contains an array that will hold the parameters to construct the system. More elegant solutions worth exploring are proposed in the cited work.

Listing 13.10: Mix-in constructor problem. What should be placed in '***' ?

```
struct Base
{
    Base(int x){};
};

template <typename Wrapped>
struct Wrapper : public Wrapped
{
    Derived(***) : Wrapped(***)
    {
    }
};

void main()
{
    Derived<Base> derived(???);
}
```

13.6.2 Synapses

We lack a general mechanism of building synapses that connect two dynamical systems, and that have dynamical behaviour themselves. The synapses provided with the library are mostly hand tailored to the task, even though some general mechanisms are already present. We will extend the framework to allow for more flexible creation of synapse classes.

Bibliography

Afraimovich, V. S., Rabinovich, M. I. and Varona, P. (2004), "Heteroclinic contours in neural ensembles and the winnerless competition principle", *International Journal of Bifurcation and Chaos [in Applied Sciences and Engineering]* , Vol. 14.

Aguirre, C., Campos, D., Pascual, P. and Serrano, E. (2005), "A model of Spiking-Bursting neuronal behavior using a piecewise linear Two-Dimensional map", *Computational Intelligence and Bioinspired Systems* , pp. 130–135.

Alexander, C., Ishikawa, S. and Silverstein, M. (1977), *A Pattern Language: Towns, Buildings, Construction (Center for Environmental Structure Series)*, later printing edn, Oxford University Press.

Aoi, S. and Tsuchiya, K. (2006), "Stability analysis of a simple walking model driven by an oscillator with a phase reset using sensory feedback", *IEEE Transactions on Robotics* , Vol. 22, pp. 391–397.

Arena, P., Fortuna, L., Frasca, M. and Patane, L. (2005), "A CNN-based chip for robot locomotion control", *IEEE Transactions on Circuits and Systems I: Regular Papers* , Vol. 52, pp. 1862–1871.

Arshavsky, Y., Grillner, S., Orlovsky, G. and Panchin, Y. (1991), *Central Generators and The Spatio-Temporal Pattern of Movements*, Vol. 81 of *Advances in Psychology*, Elsevier, pp. 93–115.

Arshavsky, Y. I., Deliagina, T. G., Orlovsky, G. N., Panchin, Y. V., Popova, L. B. and Sadreyev, R. I. (1998), "Analysis of the central pattern generator for swimming in the mollusk clione", *Annals of the New York Academy of Sciences* , Vol. 860, pp. 51–69.

Ayers, J., Wilbur, C. and Olcott, C. (2000), Lamprey robots, *in* T. Wu and N. Kato, eds, 'International Symposium on Aqua Biomechanisms', Tokai University.

Ayers, J. and Witting, J. (2007), "Biomimetic approaches to the control of underwater walking machines", *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* , Vol. 365, pp. 273–295.

Boashash, B. (1992), Estimating and interpreting the instantaneous frequency of a signal, *in* 'Proceedings of the IEEE', Vol. 80, pp. 520–538.

Brezina, V., Orekhova, I. V. and Weiss, K. R. (2000), "The neuromuscular transform: The dynamic, nonlinear link between motor neuron firing patterns and muscle contraction in rhythmic behaviors", *Journal of Neurophysiology* , Vol. 83, pp. 207–231.

Brown, G. T. (1911), "The intrinsic factors in the act of progression in the mammal", *Proceedings of the Royal Society of London* , Vol. 84 of *B*, pp. 308–319.

Bucher, D., Prinz, A. A. and Marder, E. (2005), "Animal-to-Animal variability in motor pattern production in adults and during growth", *J. Neurosci.* , Vol. 25, pp. 1611–1619.

Chung, S.-J. and Dorothy, M. (2009), "Neurobiologically inspired control of engineered flapping flight", *Journal of Guidance, Control, and Dynamics* , Vol. 33, pp. 440–453.

Cohen, A. H., Holmes, P. J. and Rand, R. H. (1982), "The nature of the coupling between segmental oscillators of the lamprey spinal generator for locomotion: a mathematical model.", *Journal of mathematical biology* , Vol. 13, pp. 345–369.

Conradt, J. and Varshavskaya, P. (2003), Distributed central pattern generator control for a serpentine robot, *in* 'In ICANN 2003'.

Crespi, A., Badertscher, A., Guignard, A. and Ijspeert, A. J. (2005), "Amphibot i: an amphibious snake-like robot", *Robotics and Autonomous Systems* , Vol. 50, pp. 163–175.

Crespi, A. and Ijspeert, A. J. (2008), "Online optimization of swimming and crawling in an amphibious snake robot", *IEEE Transactions on Robotics* , Vol. 24, pp. 75–87.

Degallier, S. and Ijspeert, A. J. (2010), "Modeling discrete and rhythmic movements through motor primitives: a review", *Biological Cybernetics* , Vol. 103, Springer Berlin / Heidelberg, pp. 319–338.

Destexhe, A., Mainen, Z. F. and Sejnowski, T. J. (1994), "An efficient method for computing synaptic conductances based on a kinetic model of receptor binding", *Neural Computation* , Vol. 6, MIT Press, Cambridge, MA, USA, pp. 14–18.

Eisenecker, U. W., Blinn, F. and Czarnecki, K. (2000), A solution to the Constructor-Problem of Mixin-Based programming in c++, *in* 'First Workshop on C++ Template Programming, Erfurt, Germany'.

Ekeberg, O. (1993), "A combined neuronal and mechanical model of fish swimming", *Biological Cybernetics* , Vol. 69, pp. 363–374.

Ekeberg, O., Grillner, S. and Lansner, A. (1995), "The neural control of fish swimming studied through numerical simulations", *Adaptive Behavior* , Vol. 3, pp. 363–384.

Elson, R. C., Selverston, A. I., Abarbanel, H. D. and Rabinovich, M. I. (2002), "Inhibitory synchronization of bursting in biological neurons: Dependence on synaptic time constant", *J Neurophysiol* , Vol. 88, pp. 1166–1176.

Gamma, E., Helm, R., Johnson, R. and Vlissides, J. (1993), Design patterns: Abstraction and reuse of Object-Oriented design, *in* O. Nierstrasz, ed., 'ECOOP'93 Object-Oriented Programming', Vol. 707 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, Berlin, Heidelberg, chapter 21, pp. 406–431–431.

Gamma, E., Helm, R., Johnson, R. and Vlissides, J. M. (1994), *Design Patterns: Elements of Reusable Object-Oriented Software*, 1 edn, Addison-Wesley Professional.

Getting, P. A. (1989), "Emerging principles governing the operation of neural networks", *Annual Review of Neuroscience* , Vol. 12, pp. 185–204.

González-Gómez, J. and Boemo Scalvinoni, E. (2008), Robótica modular y locomoción: aplicación a robots ápodos, PhD thesis, Escuela Politécnica Superior. Universidad Autónoma de Madrid.

González-Gómez, J. and Scalvinoni, E. B. (2003), Diseño de robots ápodos, Master's thesis, Escuela Politécnica Superior, Universidad Autónoma de Madrid.

González-gómez, J., Zhang, H., Boemo, E. and Zhang, J. (2006), Locomotion capabilities of a modular robot with eight Pitch-Yaw-connecting modules, *in* '9th International Conference on Climbing and Walking Robots. CLAWAR06'.

Grillner, S. (2006), "Biological pattern generation: The cellular and computational logic of networks in motion", *Neuron* , Vol. 52, Cell Press,, pp. 751–766.

Grillner, S., Hellgren, J., Ménard, A., Saitoh, K. and Wikström, M. A. (2005), "Mechanisms for selection of basic motor programs–roles for the striatum and pallidum.", *Trends Neurosci* , Vol. 28, Nobel Institute for Neurophysiology, Department of Neuroscience, Karolinska Institutet, SE 17177 Stockholm, Sweden. sten.grillner@neuro.ki.se, pp. 364–370.

Herrero-Carrón, F., Rodríguez, F. B. and Varona, P. (2010*a*), Dynamical invariants for CPG control in autonomous robots, *in* J. Filipe, J. A. Cetto and J.-L. Ferrier, eds, '7th International Conference on Informatics in Control, Automation and Robotics', Vol. 2, pp. 441–445.

Herrero-Carrón, F., Rodríguez, F. B. and Varona, P. (2010*b*), Novel modular CPG topologies for modular robotics, *in* 'IEEE 2010 International Conference on Robotics and Automation workshop "Modular Robots: The State of the Art"', IEEE, pp. 89–93.

Herrero-Carrón, F., Rodríguez, F. B. and Varona, P. (2010*c*), Studying robustness against noise in oscillators for robot control, *in* 'International conference on automation, robotics and control systems', pp. 58–63.

Herrero-Carrón, F., Rodríguez, F. B. and Varona, P. (2011), Flexible entrainment in a bio-inspired modular oscillator for modular robot locomotion, *in* J. Cabestany, I. Rojas and G. Joya, eds, 'Advances in Computational Intelligence', Vol. 6692 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, Berlin, Heidelberg, chapter 67, pp. 532–539.

Herrero-Carrón, F., Rodríguez, F. B. and Varona, P. (2011), "Bio-inspired design strategies for central pattern generator control in modular robotics", *Bioinspiration & Biomimetics* , Vol. 6, pp. 016006+.

Hindmarsh, J. L. and Rose, R. M. (1984), "A model of neuronal bursting using three coupled first order differential equations", *Proceedings Of The Royal Society Of London. Series B, Containing Papers Of a Biological Character. Royal Society (Great Britain)* , Vol. 221, pp. 87–102.

Hodgkin, A. L. and Huxley, A. F. (1952), "A quantitative description of membrane current and its application to conduction and excitation in nerve.", *The Journal of physiology* , Vol. 117, pp. 500–544.

Huang, N. E., Shen, Z., Long, S. R., Wu, M. C., Shih, H. H., Zheng, Q., Yen, N. C., Tung, C. C. and Liu, H. H. (1998), "The empirical mode decomposition and the hilbert spectrum for nonlinear and non-stationary time series analysis", *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* , Vol. 454, pp. 903–995.

Huerta, R., Varona, P., Rabinovich, M. I. and Abarbanel, H. D. (2001), "Topology selection by chaotic neurons of a pyloric central pattern generator.", *Biological Cybernetics* , Vol. 84, Springer Berlin / Heidelberg, pp. L1–L8.

Ijspeert, A. (2008), "Central pattern generators for locomotion control in animals and robots: A review", *Neural Networks* , Vol. 21, pp. 642–653.

Ijspeert, A., Crespi, A. and Cabelguen, J.-M. (2005), "Simulation and robotics studies of salamander locomotion", *Neuroinformatics* , Vol. 3, pp. 171–195.

Ijspeert, A. J. (2001), "A connectionist central pattern generator for the aquatic and terrestrial gaits of a simulated salamander", *Biological Cybernetics* , Vol. 84, Springer Berlin / Heidelberg, pp. 331–348.

Ijspeert, A. J., Crespi, A., Ryczko, D. and Cabelguen, J.-M. (2007), "From swimming to walking with a salamander robot driven by a spinal cord model", *Science* , Vol. 315, School of Computer and Communication Sciences, Ecole Polytechnique Fédérale de Lausanne (EPFL), Station 14, CH-1015 Lausanne, Switzerland. auke.ijspeert@epfl.ch, pp. 1416–1420.

Ijspeert, A. J. and Kodjabachian, J. (1999), "Evolution and development of a central pattern generator for the swimming of a lamprey", *Artificial Life* , Vol. 5, MIT Press, pp. 247–269.

Kamimura, A., Kurokawa, H., Toshida, E., Tomita, K., Murata, S. and Kokaji, S. (2003), Automatic locomotion pattern generation for modular robots, *in* 'IEEE International Conference on Robotics and Automation', Vol. 1, pp. 714–720 vol.1.

Kamimura, A., Kurokawa, H., Yoshida, E., Tomita, K., Kokaji, S. and Murata, S. (2004), Distributed adaptive locomotion by a modular robotic system, M-TRAN II, *in* 'Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on', Vol. 3, IEEE, pp. 2370–2377 vol.3.

Kopell, N. and Ermentrout, G. (2000), 'Mechanisms of phase-locking and frequency control in pairs of coupled neural oscillators'.

Kopell, N. and Somers, D. (1995), "Anti-phase solutions in relaxation oscillators coupled through excitatory interactions", *Journal of Mathematical Biology* , Vol. 33, Springer Berlin / Heidelberg, pp. 261–280.

Kristan, W. B. (2007), "A Push-Me Pull-You neural design", *Science* , Vol. 315, pp. 339–340.

Kurokawa, H., Tomita, K., Kamimura, A., Kokaji, S., Hasuo, T. and Murata, S. (2008), "Distributed Self-Reconfiguration of M-TRAN III modular robotic system", *International Journal of Robotics Research* , Vol. 27, Sage Publications, Inc., Thousand Oaks, CA, USA, pp. 373–386.

Latorre, R., Rodríguez, F. B. and Varona, P. (2006), "Neural signatures: multiple coding in spiking-bursting cells.", *Biological Cybernetics* , Vol. 95, pp. 169–183.

Liu, G., Habib, M., Watanabe, K. and Izumi, K. (2008), "Central pattern generators based on matsuoka oscillators for the locomotion of biped robots", *Artificial Life and Robotics* , Vol. 12, pp. 264–269.

Manoonpong, P., Geng, T., Kulvicius, T., Porr, B. and Wörgötter, F. (2007), "Adaptive, fast walking in a biped robot under neuronal control and learning", *PLoS Computational Biology* , Vol. 3, pp. 1305–1320.

Marbach, D. and Ijspeert, A. J. (2005), Online optimization of modular robot locomotion, *in* 'Mechatronics and Automation, 2005 IEEE International Conference', Vol. 1, IEEE, pp. 248–253 Vol. 1.

Marder, E. and Bucher, D. (2001), "Central pattern generators and the control of rhythmic movements.", *Current biology* , Vol. 11, Volen Center, MS 013, Brandeis University, 415 South Street, Waltham, Massachusetts 02454-9110, USA. marder@brandeis.edu.

Marder, E. and Bucher, D. (2007), "Understanding circuit dynamics using the stomatogastric nervous system of lobsters and crabs", *Annual Review of Physiology* , Vol. 69, Volen Center and Biology Department, Brandeis University, Waltham, MA 02454 marder@brandeis.edu., pp. 291–316.

Marder, E. and Goaillard, J.-M. (2006), "Variability, compensation and homeostasis in neuron and network function", *Nature Reviews Neuroscience* , Vol. 7, Nature Publishing Group, pp. 563–574.

Marder, E. and Prinz, A. A. (2002), "Modeling stability in neuron and network function: the role of activity in homeostasis.", *BioEssays : news and reviews in molecular, cellular and developmental biology.* , Vol. 24, Volen Center, Brandeis University, Waltham, MA 02454-9110, USA. marder@brandeis.edu, pp. 1145–1154.

Matsuo, T. and Ishii, K. (2007), "A CPG control system for a modular type mobile robot", *International Congress Series* , Vol. 1301, pp. 206–209.

Matsuoka, K. (1987), "Mechanisms of frequency and pattern control in the neural rhythm generators", *Biological Cybernetics* , Vol. 56, pp. 345–353.

Murata, S., Yoshida, E., Kamimura, A., Kurokawa, H., Tomita, K. and Kokaji, S. (2002), "M-TRAN: self-reconfigurable modular robotic system", *Mechatronics, IEEE/ASME Transactions on* , Vol. 7, pp. 431–441.

Oppenheim, A. V., Schafer, R. W. and Buck, J. R. (1999), *Discrete-Time Signal Processing (2nd Edition) (Prentice-Hall Signal Processing Series)*, 2 edn, Prentice Hall, chapter 11, pp. 775–802.

Oprisan, S. A., Prinz, A. A. and Canavier, C. C. (2004), "Phase resetting and phase locking in hybrid circuits of one model and one biological neuron.", *Biophys J* , Vol. 87, Department of Psychology, University of New Orleans, New Orleans, Louisiana 70148, USA. soprisan@uno.edu, pp. 2283–2298.

Prinz, A. A., Bucher, D. and Marder, E. (2004), "Similar network activity from disparate circuit parameters", *Nature Neuroscience* , Vol. 7, Nature Publishing Group, pp. 1345–1352.

Rabinovich, M. I., Varona, P., Selverston, A. I. and Abarbanel, H. D. I. (2006), "Dynamical principles in neuroscience", *Reviews of Modern Physics* , Vol. 78, American Physical Society, pp. 1213–1265.

Reyes, M., Huerta, R., Rabinovich, M. and Selverston, A. (2008), "Artificial synaptic modification reveals a dynamical invariant in the pyloric CPG", *European Journal of Applied Physiology* , Vol. 102, pp. 667–675.

Rowat, P. F. and Selverston, A. I. (1997), "Oscillatory mechanisms in pairs of neurons connected with fast inhibitory synapses", *Journal of Computational Neuroscience* , Vol. 4, Springer Netherlands, pp. 103–127.

Rulkov, N. F. (2002), "Modeling of spiking-bursting neural behavior using two-dimensional map", *Physical Review E* , Vol. 65, American Physical Society, pp. 041922+.

Selverston, A. B., Szücs, A., Huerta, R., Pinto, R. D. and Reyes, M. (2009), "Neural mechanisms underlying the generation of the lobster gastric mill motor pattern", *Frontiers in Neural Circuits* .

Selverston, A. I. and Moulins, M., eds (1987), *The Crustacean Stomatogastric System: A Model for the Study of Central Nervous Systems*, 1 edn, Springer.

Selverston, A. I., Rabinovich, M. I., Abarbanel, H. D. I., Elson, R., Szucs, A., Pinto, R. D., Huerta, R. and Varona, P. (2000), "Reliable circuits from irregular neurons: A dynamical approach to understanding central pattern generators", *Journal of Physiology-Paris* , Vol. 94, pp. 357–374.

Seo, K., Chung, S.-J. and Slotine, J. J. (2010), "CPG-based control of a turtle-like underwater vehicle", *Autonomous Robots* , Vol. 28, Kluwer Academic Publishers, Hingham, MA, USA, pp. 247–269.

Shilnikov, A. L. and Rulkov, N. F. (2003), "Origin of chaos in a two-dimensional map modeling spike-bursting neural activity", *International Journal of Bifurcation and Chaos* , Vol. 13, pp. 3325–3340.

Siek, J. and Lumsdaine, A. (2000), Concept checking: Binding parametric polymorphism in c++, *in* 'Proceedings First Workshop on C++ Template Programming', Erfurt, Germany.

Smaragdakis, Y. and Batory, D. (2001), Mixin-Based programming in c++, *in* G. Butler and S. Jarzabek, eds, 'Generative and Component-Based Software Engineering', Vol. 2177 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, Berlin, Heidelberg, chapter 12, pp. 164–178.

Smarandache, C., Hall, W. M. and Mulloney, B. (2009), "Coordination of Rhythmic Motor Activity by Gradients of Synaptic Strength in a Neural Circuit That Couples Modular Neural Oscillators", *Journal of Neuroscience* , Vol. 29, pp. 9351–9360.

Sproewitz, A., Moeckel, R., Maye, J. and Ijspeert, A. J. (2008), "Learning to move in modular robots using central pattern generators and online optimization", *The International Journal of Robotics Research* , Vol. 27, pp. 423–443.

Stiesberg, G. R., Reyes, M. B., Varona, P., Pinto, R. D. and Huerta, R. (2007), "Connection topology selection in central pattern generators by maximizing the gain of information", *Neural Computation* , Vol. 19, MIT Press, pp. 974–993.

Szucs, A., Pinto, R. D., Rabinovich, M. I., Abarbanel, H. D. and Selverston, A. I. (2003), "Synaptic modulation of the interspike interval signatures of bursting pyloric neurons.", *Journal of Neurophysiology* , Vol. 89, pp. 1363–1377.

Thoby-Brisson, M. and Simmers, J. (1998), "Neuromodulatory inputs maintain expression of a lobster motor Pattern-Generating network in a Modulation-Dependent state: Evidence from Long-Term decentralization in vitro", *J. Neurosci.* , Vol. 18, pp. 2212–2225.

Thuma, J. B., Morris, L. G., Weaver, A. L. and Hooper, S. L. (2003), "Lobster (panulirus interruptus) pyloric muscles express the motor patterns of three neural networks, only one of which innervates the muscles", *Journal of Neuroscience* , Vol. 23, pp. 8911–8920.

von Bertalanffy, L. (2006), *General System Theory: Foundations, Development, Applications*, revised edn, George Braziller.

Wang, X.-J. and Rinzel, J. (1992), "Alternating and synchronous rhythms in reciprocally inhibitory model neurons", *Neural Comput.* , Vol. 4, MIT Press, Cambridge, MA, USA, pp. 84–97.

Yim, M. (1994), Locomotion with a unit-modular reconfigurable robot, PhD thesis, Stanford University.

Yim, M., Duff, D. G. and Roufas, K. D. (2000), PolyBot: a modular reconfigurable robot, *in* 'Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on', Vol. 1, pp. 514–520 vol.1.

Zhurov, Y. and Brezina, V. (2006), "Variability of motor neuron spike timing maintains and shapes contractions of the accessory radula closer muscle of aplysia", *Journal Neuroscience* , Vol. 26, pp. 7056–7070.